Practical Electronics

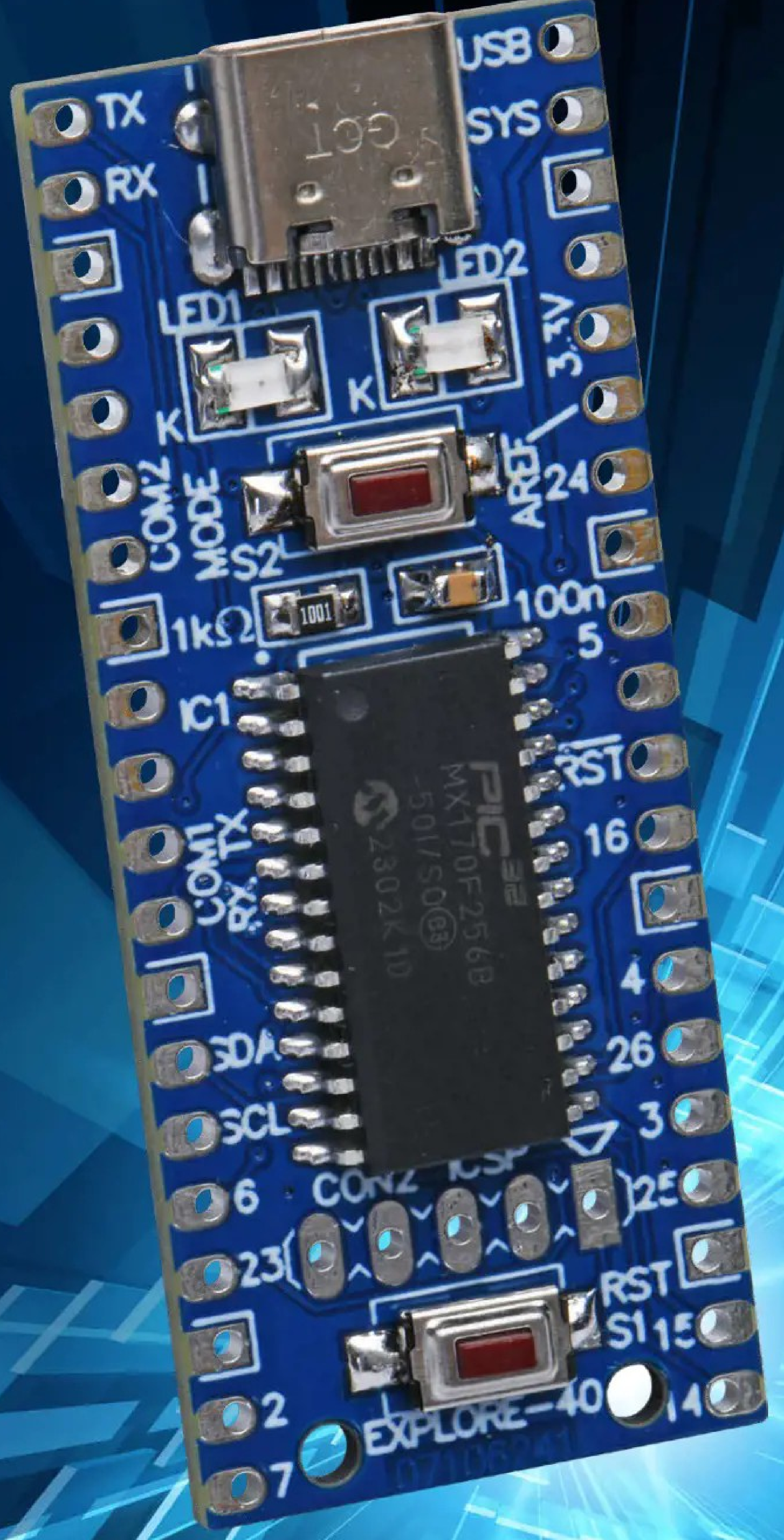
The UK's premier electronics, computing and maker magazine

Circuit Surgery
Getting our DSP board's ADC & DAC working together

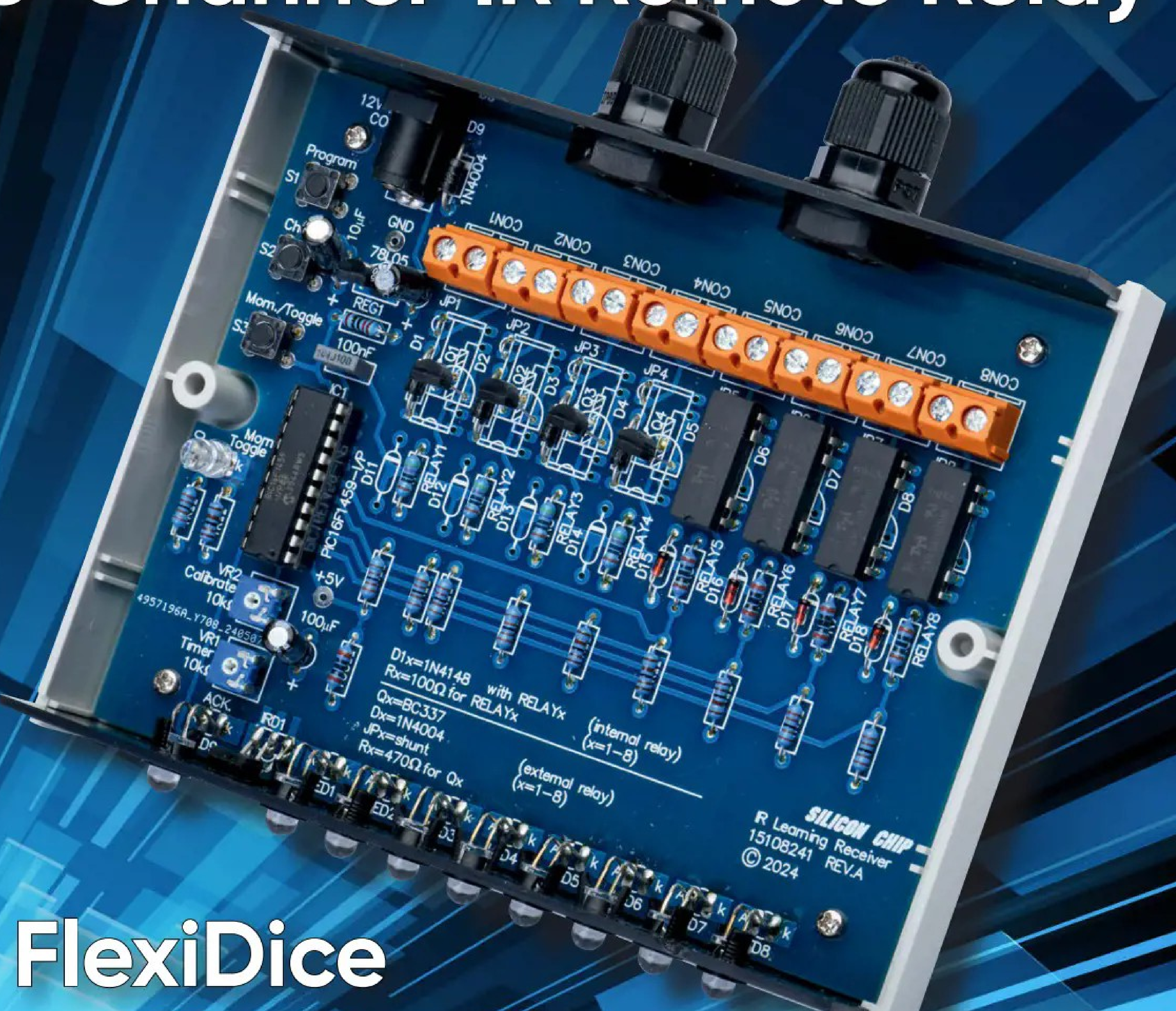
Audio Out
Minimising distortion in balanced input transformers

Precision Electronics
Choosing & using voltage references

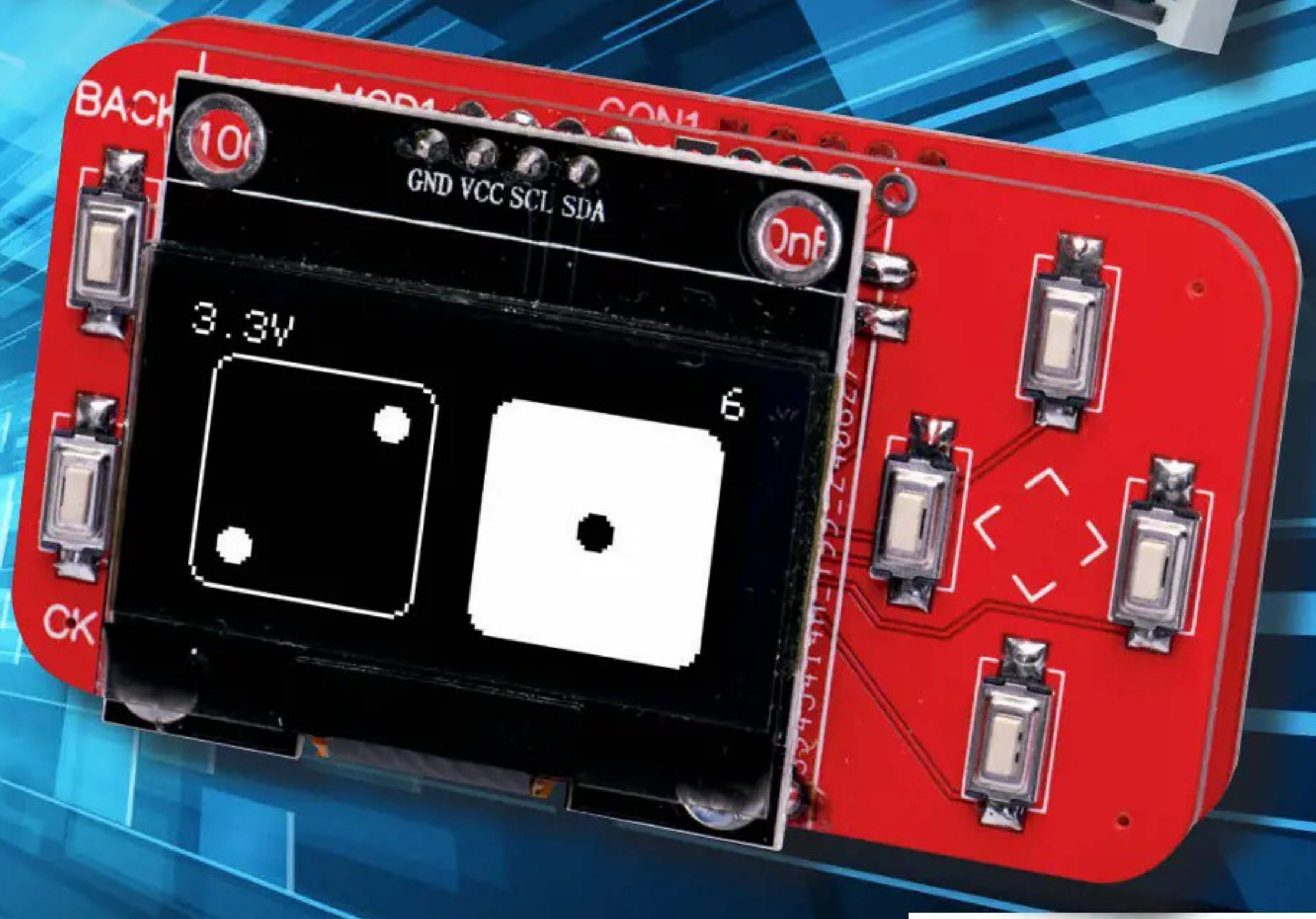
Explore-40 MMBasic in a Pico form factor



8-Channel IR Remote Relay


SILICON CHIP
IR Learning Receiver
15108241 REV.A
© 2024

FlexiDice


GND VCC SCL SDA
3.3V
6

PLUS!

Techno Talk – finding chip data & pseudo intelligence
Cool Beans – ultimate switch debouncing and RNG
MIPI's new I3C serial standard can replace both I²C & SPI

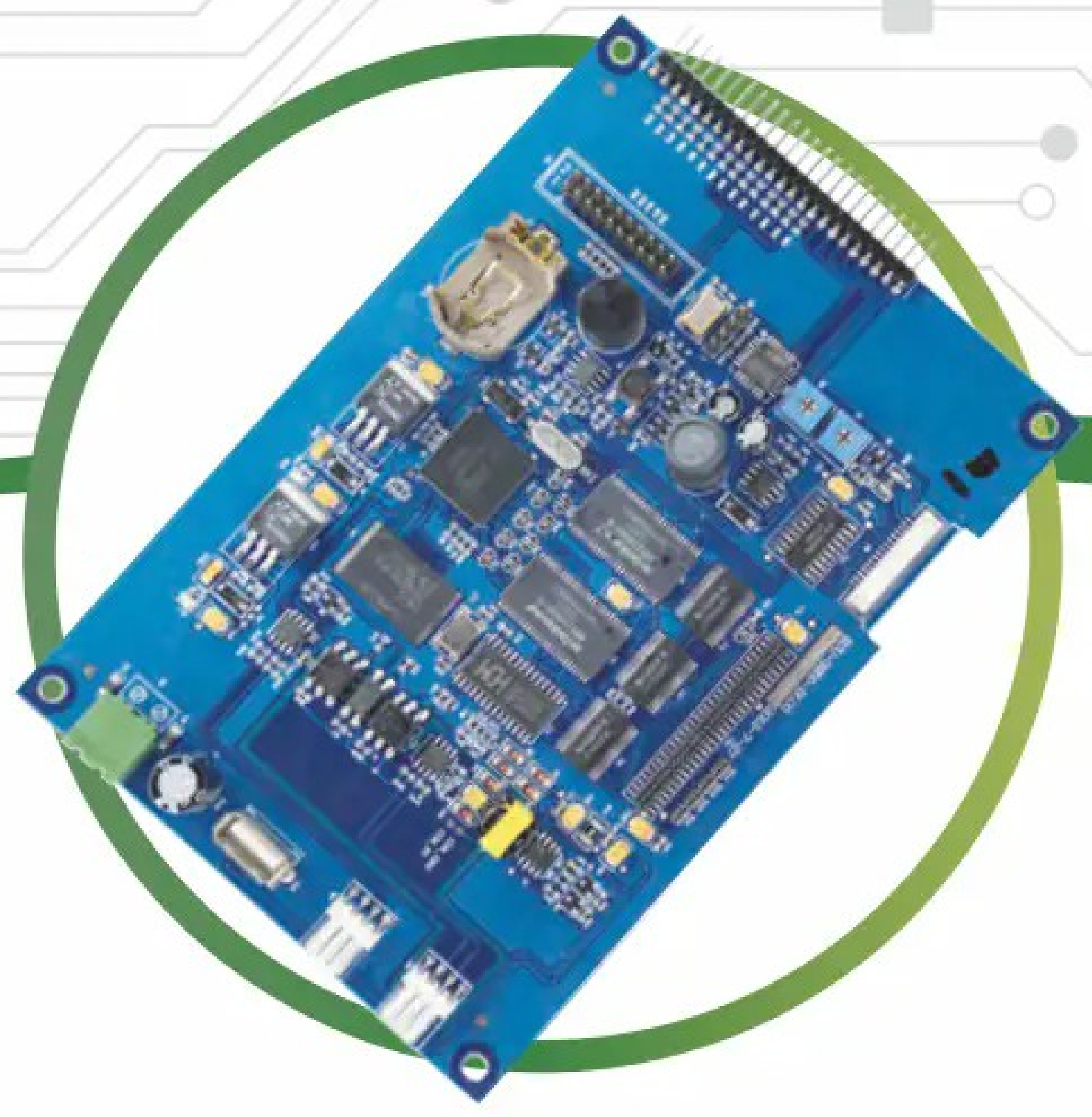
Aug 2025 £5.99
9 772632 573030 08

www.electronpublishing.com | @practicalelec | practicalelectronics

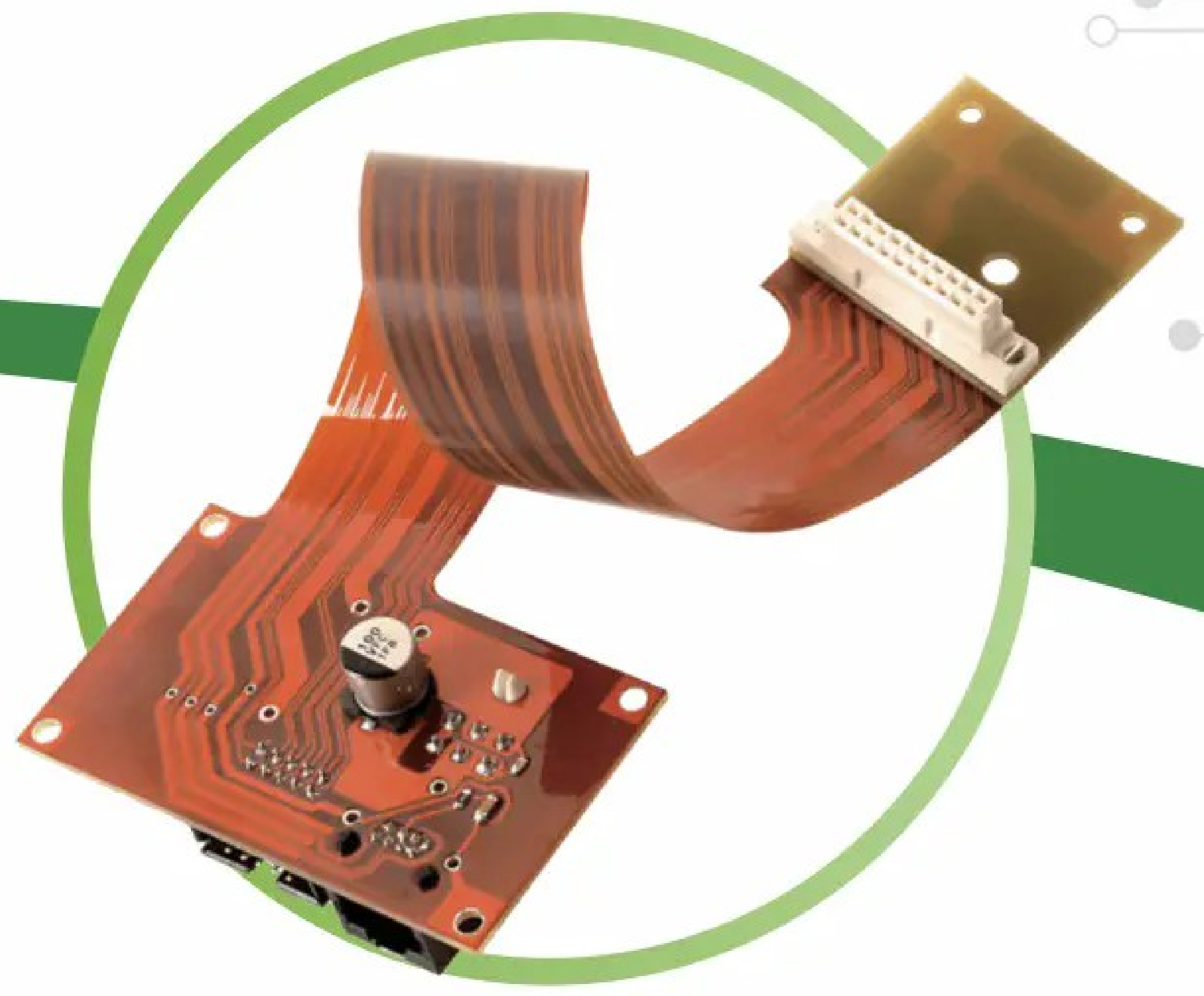
FULL-SERVICE
ELECTRONICS
MANUFACTURER

PCBWay

PCB Capabilities:

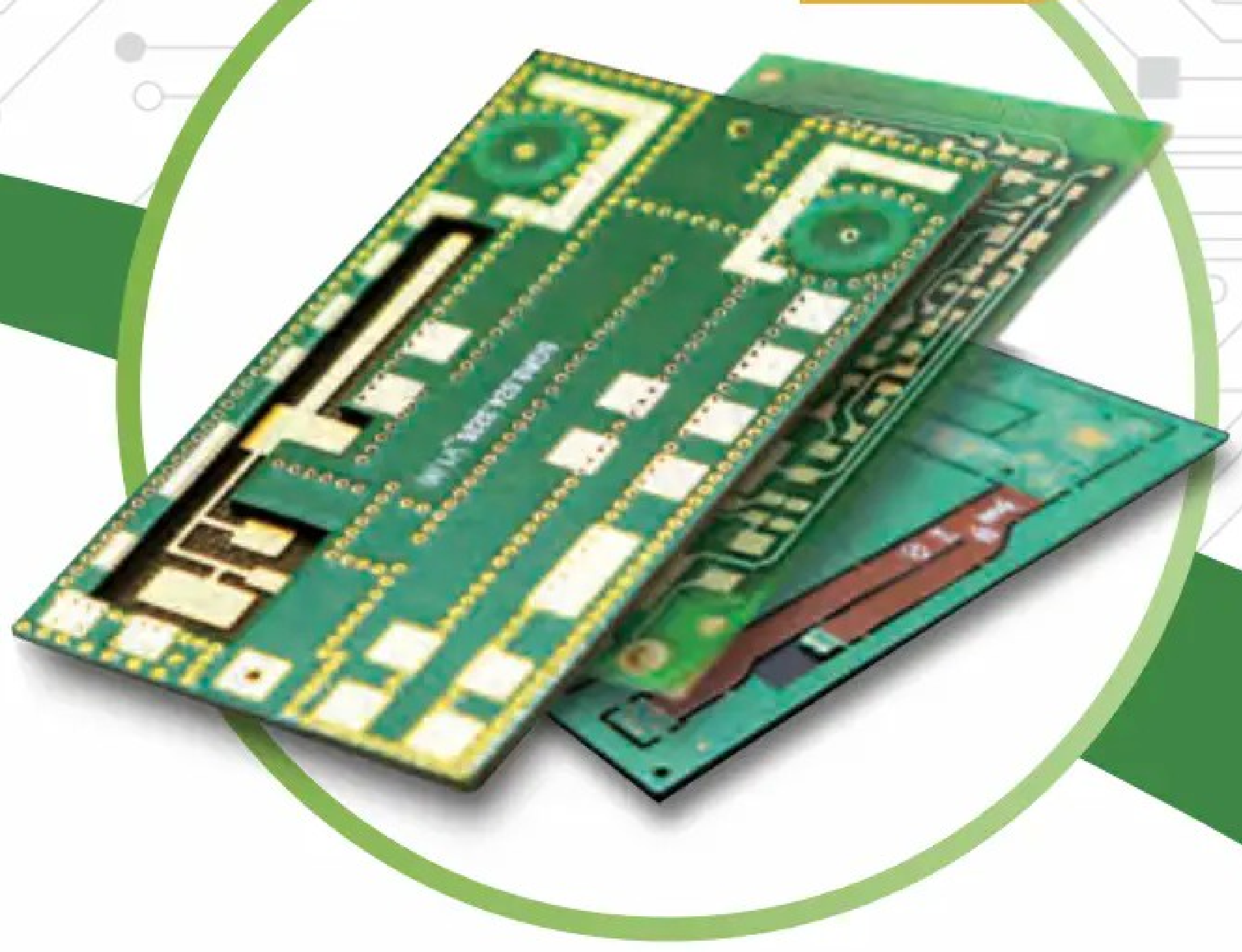


PCB Assembly
From $29
- Improved assembly precision
- Short turnaround for assembly
- Flexible support



Flex, Rigid-Flex PCB
From $46.74
- Up to 16 layers Flex circuits
- Polymide Flex, PET, High Frequency
- 0.025-≥0.4 mm FPC thickness

Special price!



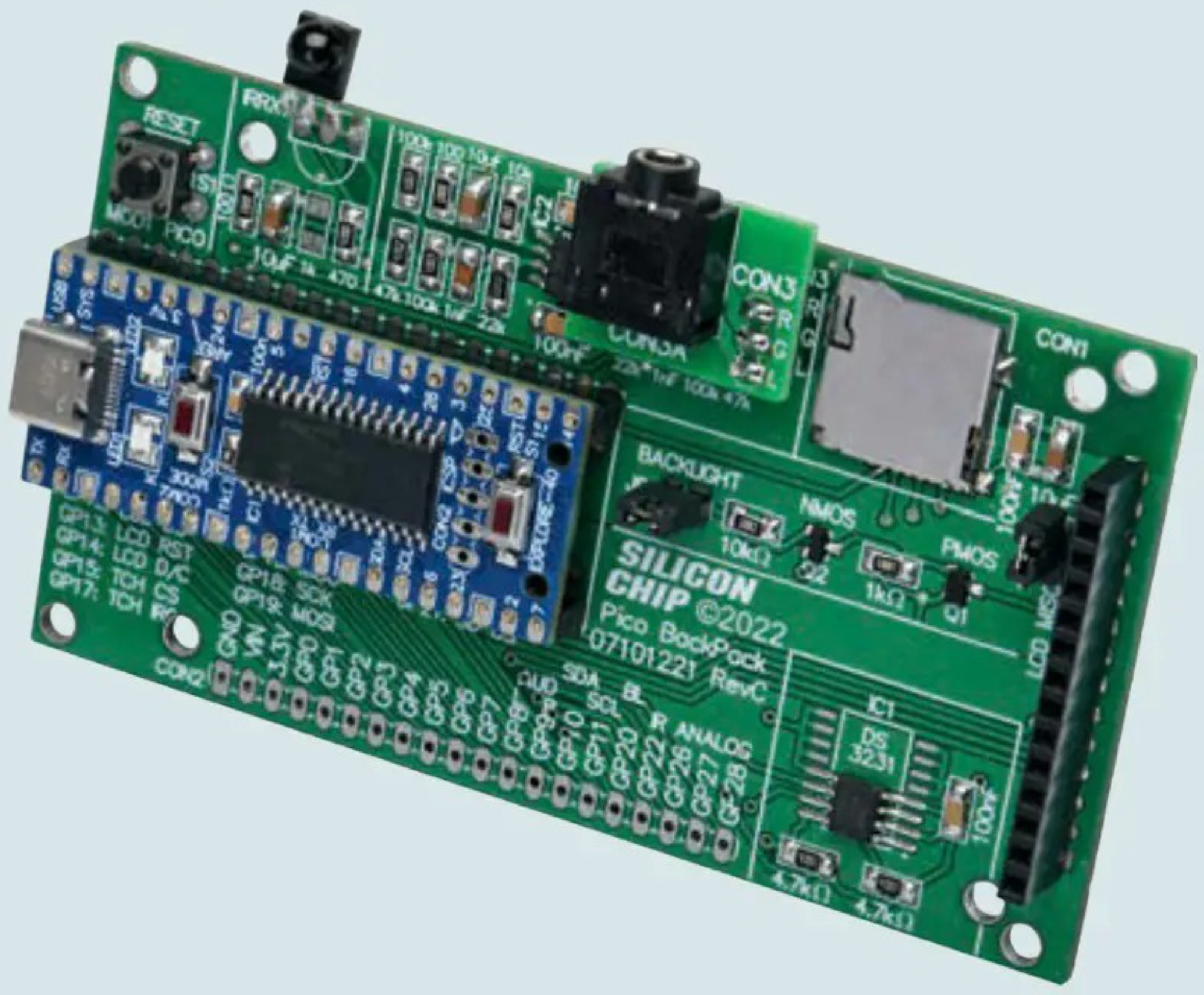
PCB Prototype
From $5/10 pcs
- Multiple color options
- 24-hour production time
- Worldwide delivery

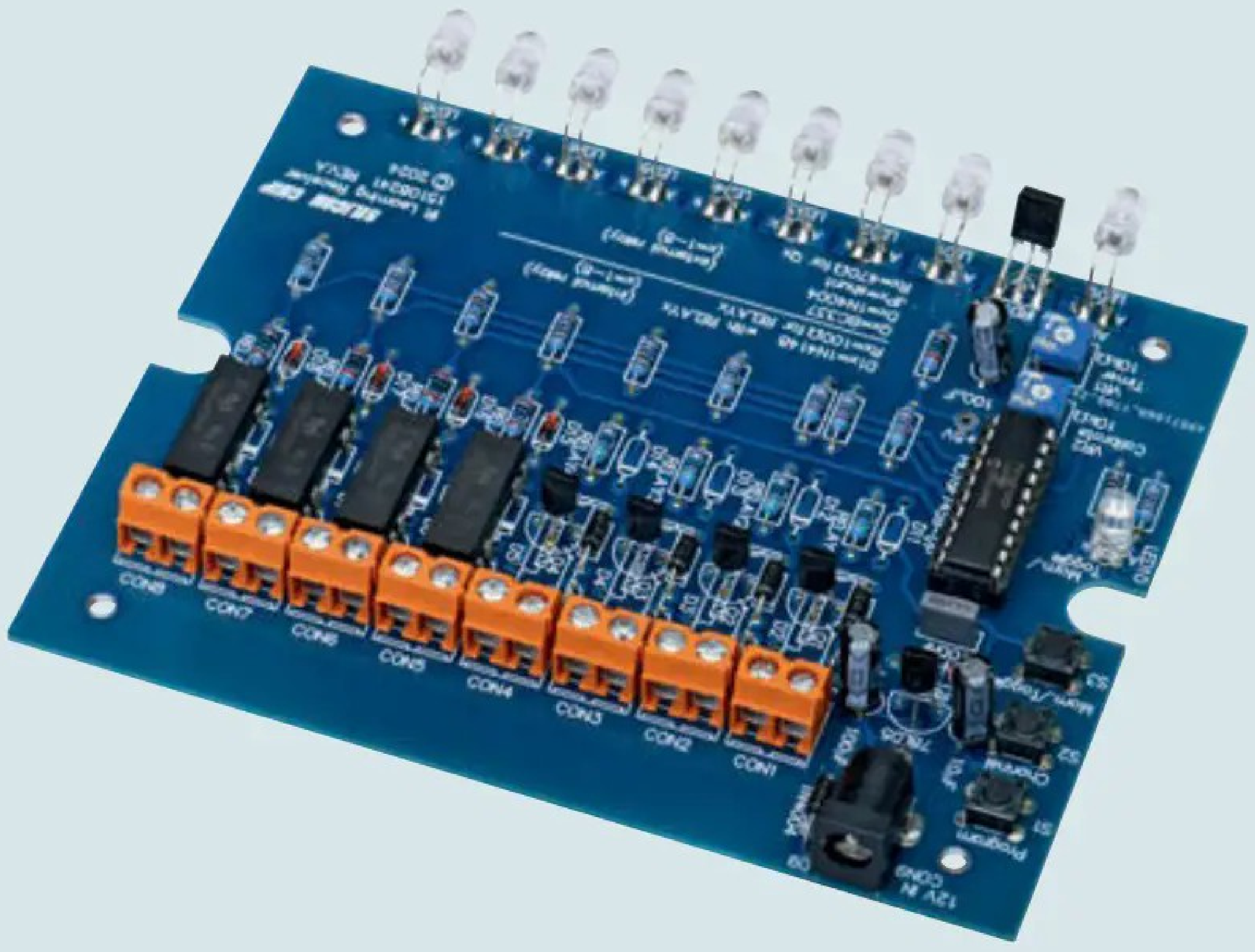
www.pcbway.com
service@pcbway.com

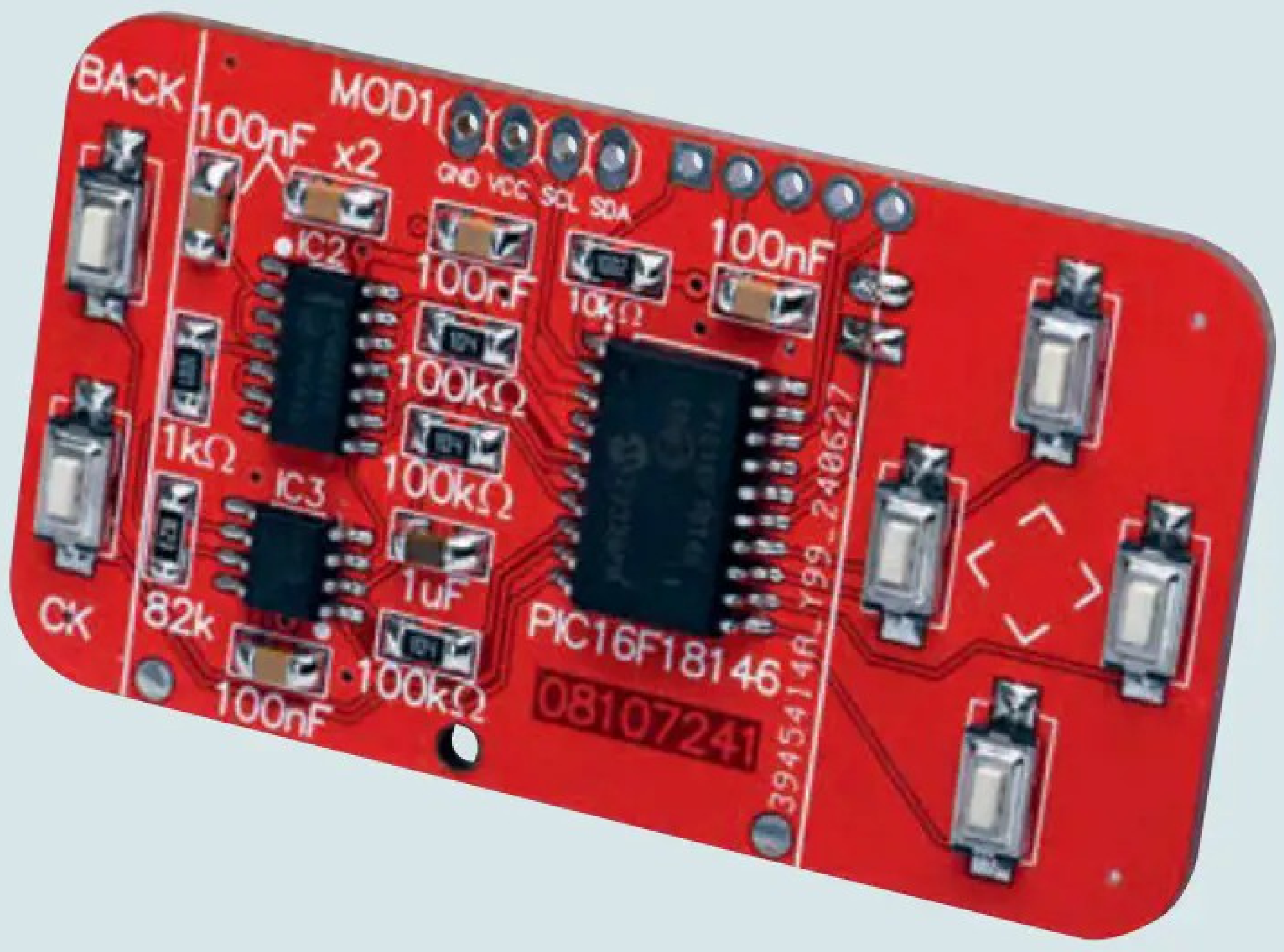
Practical
Electronics

Volume 54. No. 8
August 2025
ISSN 2632 573X

Contents







Projects and Circuits

The Micromite Explore-40 by Tim Blythman 6
A wealth of software has been written for the Micromite in BASIC. This compact Explore-40 board is a Micromite in the same form factor as the popular Pico boards, allowing a Micromite to be used with hardware designed for the Pico.

8-Channel Learning IR Remote Receiver by John Clarke 22
This eight-channel relay board can be controlled by universal remotes. Each output can be set to toggle on or off, switched on for a fixed period, or on while the button is held down. It also supports external relays, to handle higher currents or voltages.

FlexiDice by Tim Blythman 44
We've published several dice projects over the years but all have been traditional six-sided affairs. Other types of dice are used for various games and activities; this dice project can emulate dice with up to 100 faces, coin tossing and other random events.

180-230V DC Motor Speed Controller, part two by John Clarke 62
Our new High-voltage DC Motor Speed Controller, revealed last month, can control motors commonly used in lathes and treadmills, from stopped up to full speed. This article covers the assembly, testing and setup of the Speed Controller.

Series, Features and Columns

Techno Talk by Max The Magnificent 4
Over the past few months, I've been introduced to a smorgasbord of pseudo-intelligence (PI) enabled tools and technologies, some of which I'd like to share with you.

MIPI I3C by Andrew Levido 14
For many years, I^{2}C has been the best solution for low-data-rate control signalling, and SPI for higher data rates. The new I3C standard combines the best of both.

Circuit Surgery by Ian Bell 32
Topics in digital signal processing – Implementing DSP on a microcontroller, part 3.

Audio Out by Jake Rothman 40
An interesting trick by Peter Baxandall to minimise transformer-induced distortion.

Max's Cool Beans by Max The Magnificent 54
Weird & Wonderful Arduino Projects, part 8: handling switches like heroes.

The Fox Report by Barry Fox 60
The UK's TV Future: Linear, Digital or Just Confused?

Precision Electronics, part eight by Andrew Levido 72
Last month, we looked at sampling and aliasing in DACs (digital-to-analog converters) and ADCs (analog-to-digital converters). Now we will describe how voltage references work, as they are critical to the precision of both ADCs and DACs.

Regulars and Services

Editorial 2
Environmental concerns with semiconductor manufacturing

Reader services – Editorial and Advertising Departments 2

Subscribe to Practical Electronics and save money 3

Practical Electronics Back Issue DOWNLOADS – 2024 now available! 21

Teach-in Bundle – what a bargain! 53

Practical Electronics PCB Service 76
PCBs for Practical Electronics projects

Classified ads and Advertiser index 79

Next month! – highlights of our next issue of Practical Electronics 80

Made in Australia & the UK.

Written in Britain, Australia, the USA and Ireland.

Read everywhere.

© Silicon Chip Publications Pty Ltd 2025

Copyright in all drawings, photographs, articles, technical designs, software and intellectual property published in *Practical Electronics* is fully protected, and reproduction or imitation in whole or in part are expressly forbidden.

The September 2025 issue of *Practical Electronics* will be published on Tuesday, 5th of August 2025 (see p80).

Renewals paid by PayPal, Visa and Mastercard are now handled via our website:
www.electronpublishing.com

Practical Electronics

Volume 54. No. 8
August 2025
ISSN 2632 573X

Editorial

Editorial offices
Electron Publishing (Australia)
Tel +61 2 9939 3295
Email pe@pemag.au
Web www.electronpublishing.com

Address mail to:
Electron Publishing (Australia)
PO Box 194, Matraville NSW 2036
Australia

Advertising enquiries
+61 2 9939 3295 pe@pemag.au

Editor	Nicholas Vinen
Publisher	Nicholas Vinen
Digital subscriptions	Stewart Kearn **Tel** 07918 614662
Online Editor	Alan Winstanley
Web Systems	Kris Thain
Production	Bao Smith
Technical staff	Tim Blythman, John Clarke

Print subscriptions
Practical Electronics Subscriptions
PO Box 6337
Bournemouth BH1 9EH
United Kingdom
Tel 01202 087631
Email pesubs@selectps.com

Technical enquiries

We regret that technical enquiries cannot be answered over the telephone. We are unable to offer any advice on the use, purchase, repair or modification of commercial equipment or the incorporation or modification of designs published in the magazine.

Questions about articles or projects should be sent to the editor by email: pe@pemag.au

Projects and circuits

All reasonable precautions are taken to ensure that the advice and data given to readers is reliable. We cannot, however, guarantee it and we cannot accept legal responsibility for it.

Some projects and circuits published in *Practical Electronics* employ voltages that can be lethal. Do not build, test, modify or fix any mains-powered equipment unless you fully understand the safety aspects involved and you use an RCD (GFCI) adaptor.

Component supplies

Silicon Chip Publications may offer kits or other parts for making our projects, but not in all cases. When kits are not available, readers will need to find and source parts themselves. We advise readers to check that all parts are still available before commencing any project in a back-dated issue.

Advertisements

Although the proprietors and staff of *Practical Electronics* take reasonable precautions to protect the interests of readers by ensuring as far as practicable that advertisements are bona fide, the magazine and its publishers cannot give any undertakings in respect of statements or claims made by advertisers, whether these advertisements are printed as part of the magazine, or in inserts. The Publishers regret that under no circumstances will the magazine accept liability for non-receipt of goods ordered, or for late delivery, or for faults in manufacture.

Transmitters/bugs/telephone equipment

We advise readers that certain items of radio transmitting and telephone equipment which may be advertised in our pages cannot be legally used in the UK. Readers should check the law before buying any transmitting or telephone equipment, as a fine, confiscation of equipment and/or imprisonment can result from illegal use or ownership. The laws vary from country to country; readers should check local laws.

Environmental concerns with semiconductor manufacturing

You may have heard that semiconductor manufacturing (including for devices like solar panels) uses large amounts of fresh water. If this wastewater is not properly treated, it can become contaminated with the acids and other chemicals used during processing.

The good news is that many of these devices are manufactured in China, where environmental controls have improved significantly in recent years. Air pollution levels have dropped dramatically, and it's reasonable to assume that water pollution is also being addressed.

I'm far from a chemophobe; my usual retort when someone complains that something "contains chemicals" is that we are **made** of chemicals. But there's one group of chemicals used in semiconductor manufacturing that does raise serious concern: per- and polyfluoroalkyl substances (PFAS).

There are two major problems with this group: they are highly toxic, with studies showing dramatically increased risks of certain cancers at concentrations as low as a few parts per billion (ppb); and they are extremely persistent. In the environment, they do not break down; at least not easily and, in many cases, not at all.

In fact, so many PFAS have been released into the environment (mainly from Teflon production and their use in firefighting foams) that they've contaminated water supplies worldwide. Once absorbed by the body, they tend to accumulate, as natural metabolic processes cannot break them down or flush them out.

In semiconductor manufacturing, PFAS are primarily used as surfactants in photoresists and anti-reflective coatings (ARCs). They help control surface tension, improve coating uniformity and enhance pattern resolution. PFAS are also used in plasma etching, post-etch cleaning, and resist stripping, where they aid in wetting fine features and preventing contamination or watermarking.

In addition, fluoropolymers (a PFAS subclass) are used as non-stick, chemically resistant coatings on process tools, wafer carriers, valves, pipes, and gaskets – where resistance to strong acids, solvents, and heat is critical.

Given their toxicity and near-infinite persistence, I hope safer alternatives to PFAS can be developed and adopted soon. If their use is unavoidable, strict measures must be in place to prevent environmental release, with effective recycling or degradation methods applied after use.

Nicholas Vinen, Electron Publishing (Australia)*
Publisher & Editor,
Practical Electronics Magazine

* a division of Silicon Chip Publications Pty Ltd.

Practical Electronics

UK readers SAVE £1 on every issue

SUBSCRIBE NOW!

01202 087631

Take out a one-year subscription and save more than £10 over the year.
Even better – save over £1 per issue if you subscribe for two years – a total saving of £26.

Overseas rates represent exceptional value

You also:
- **Avoid any cover price increase for the duration of your subscription**
- **Get your magazine delivered to your door each month**
- **Guarantee a copy, even if your local newsagent sells out**

Order by:
- **Phone or post with a cheque, postal order or credit card**
- **Or order online at: www.electronpublishing.com**

Practical Electronics

SUBSCRIPTION PRICES

Subscriptions for delivery direct to any address in:

UK: 6-months £33.99, 12-months £59.99, 24-months £114.99

Europe Airmail: 6-months £38.99, 12-months £69.99, 24-months £129.99

Rest Of The World Airmail: 6-months £46.99, 12-months £84.99, 24-months £164.99

Cheques (in **£ sterling only**) payable to ***Select Publisher Services Ltd*** and sent to: Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Tel: 01202 087631
Email: pesubs@selectps.com

Also via our online shop at: **www.electronpublishing.com**

Subscriptions start with the next available issue. We accept MasterCard or Visa.
(For past issues see the Back Issues page.)

DIGITAL SUBSCRIPTIONS

Electronic subscriptions are available from £29.99 for 12 months, more details at: **www.electronpublishing.com**

SUBSCRIPTION ORDER FORM

☐ 6 Months: UK £33.99, Europe £38.99 (Airmail), Rest of the World £46.99 (Airmail)
☐ 1 Year: UK £59.99, Europe £69.99 (Airmail), Rest of the World £84.99 (Airmail)
☐ 2 Years: UK £114.99, Europe £129.99 (Airmail), Rest of the World £164.99 (Airmail)

To: **Practical Electronics Subscriptions**
PO Box 6337
Bournemouth BH1 9EH
United Kingdom

Tel: 01202 087631
Email: pesubs@selectps.com

I enclose payment of £ ....
(cheque/PO in £ sterling only)
payable to ***Select Publisher Services Ltd***

☐ Please charge my Visa/Mastercard

My card number is: ....
Please print clearly, and check that you have the number correct

Card Security Code .... Valid From Date.......................
(The last 3 digits on or just under the signature strip)

Card Ex. Date ....

Name ....

Address ....

Post code .... Tel. ....

Email ....

Subscriptions start with the next available issue.

AI Drools, PI Rules

Max the Magnificent

Have you heard of pseudo-intelligence (PI)? Over the past few months, I've been introduced to a smorgasbord of PI-enabled tools and technologies, some of which I'd like to share with you here.

I hail from Yorkshire, which, for our international readers, is widely acknowledged to be the best county in England. (Don't listen to anyone from other counties who says otherwise; they are foolish folk and you cannot trust them.)

Increasingly, I find myself identifying with the characters in Monty Python's classic Four Yorkshiremen sketch (you can watch the video at https://youtu.be/ue7wM0QC5LE). Especially the part at the end where they say, "…you try an' tell the young people of today that, and they won't believe you…"!

When I tell young engineers that I spent my formative years as an engineer in a world without the internet, they look at me as though I'm a time traveller who has just dropped in from the Victorian era.

A wealth of information

I can hardly believe the wealth of information that's now available at our fingertips. These days, for example, I can locate and access data sheets for virtually any electronic component on the web in a matter of seconds.

Things weren't always this way. Prior to the introduction of the internet, engineers maintained their own mini-libraries in their offices. If you were lucky, the company you worked for might boast its own library. Often, you were required to contact local distributors and request that they share data books and application notes for components of interest.

Even after the internet impinged itself on our collective consciousness circa the early 1990s, there was little information of use on it. Even when useful information did begin to wend its way onto the system circa the mid-1990s, that information was not easy to find.

Yahoo started life as "Jerry and David's Guide to the World Wide Web" in 1994, but it was rather ropy in the early days. Google, which began as a search engine called BackRub in 1996, was significantly better, but nowhere near as impressive as it is today.

An engineers' delight

All the above leads us to 1998, when a young engineer called Randy Sargent became frustrated with the time and effort he was devoting to rifling through component vendors' websites, desperately trying to glean information on devices for use in his designs.

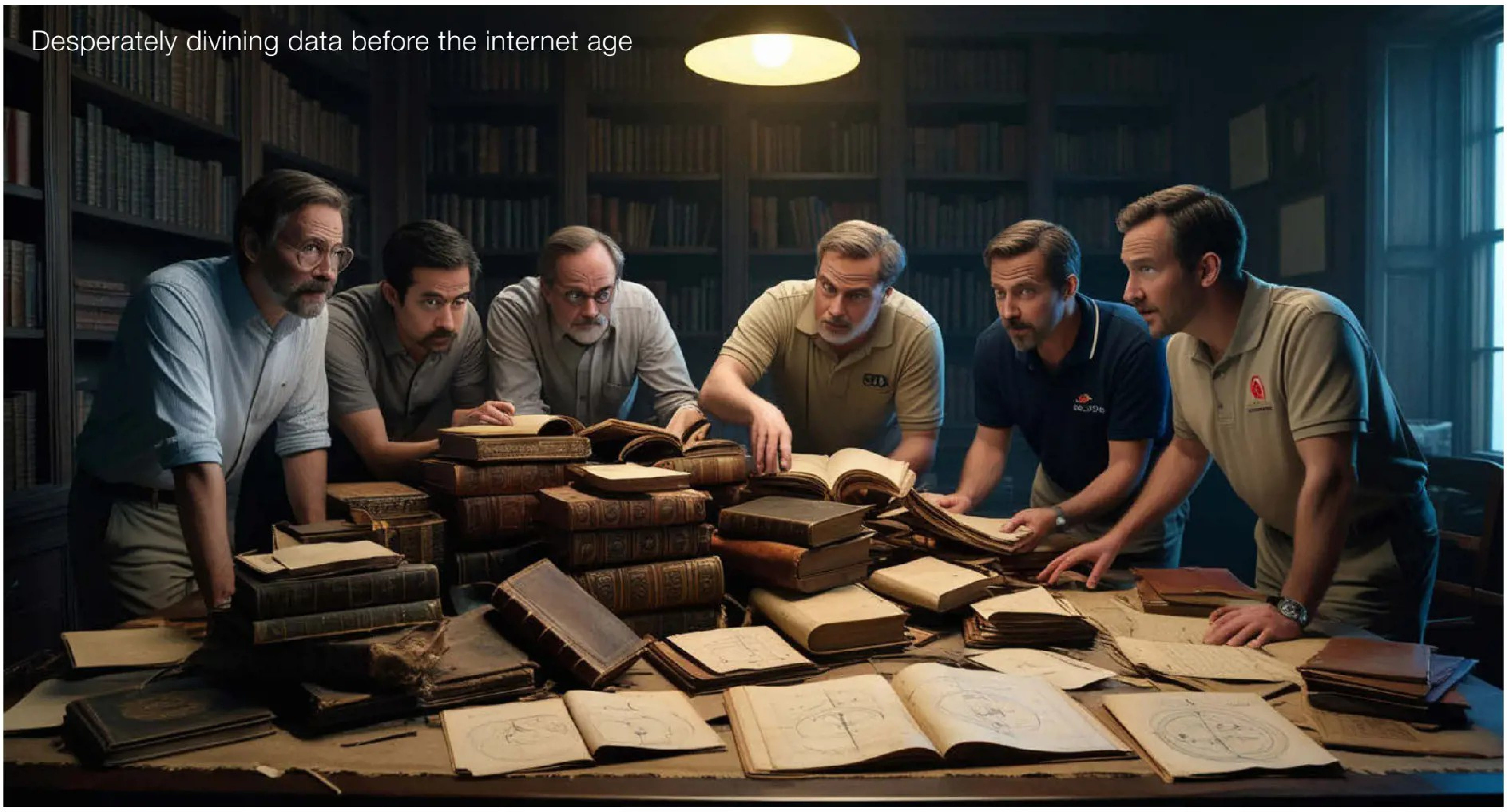
Eventually, Randy implemented an engineering hack. He created a Perl script to search the internet for him. This script could scrape component vendors' websites, extract the desired information, and present it to Randy in a form that was useful to him.

Randy's script was so successful that it evolved into FindChips (www.findchips.com), which, in addition to being both easy and free to use, is one of the most helpful component search engines I am aware of.

Pseudo intelligence

Our esteemed editor at Practical Electronics has informed me that he's tired of everyone discussing artificial intelligence (AI) all the time. You can only imagine my despondency when I discovered that I would have to find something else to write about!

Well, "you can't fight the man", as they say. Instead, I wish to share some interesting information about something we shall call pseudo-intelligence (PI) which, by some strange quirk of fate, encompasses all aspects of AI (and then some).


Desperately divining data before the internet age

PI-enabled software design tools

On the software side, we have a code completion and generation tool called Copilot, developed by GitHub (**github.com**) and the creators of ChatGPT. This is complemented by a code verification tool called Metabob, developed by the company Metabob (**metabob.com**).

Additionally, there is a documentation generator called Driver, developed by a company of the same name (**driver.ai**). This can examine the software aspects of your product, including software development kits (SDKs) and application programming interfaces (APIs), and automatically generate user documentation that you'd be proud to show your mother.

When it comes to adding PI capabilities to your embedded system, a company called DeGirum (**degirum.ai**) offers a "model zoo" of PI models for various applications, including people detection, pose detection, face detection, age estimation, gender classification, emotion classification, gesture recognition and more.

You can quickly and easily combine and deploy these models in your system.

PI-enabled hardware design

Let's start with the circuit and schematic portion of the design process. In this case, three companies immediately come to mind: Celus (**celus.io**), Circuit Mind (**circuitmind.io**), and Flux (**flux.ai**).

Celus and Flux both feature their own generative PI copilots. These allow you to do all sorts of things, like using a natural language interface to say what you want to do ("Create a WiFi-enabled thermostat using a temperature sensor and an 8-bit microcontroller") and receive a corresponding schematic in seconds.

You can also use a drag-and-drop interface to create a block diagram of your desired system (eg, a sensor, microcontroller, or motor driver), attach attributes to the blocks (eg, an I^2C interface, 16-bit data bus, or 3.3V supply) and – once again – receive a detailed schematic, including all ancillary components like resistors, capacitors and inductors, in seconds.

Celus recently added the ability to input an image of a block diagram that you created on the back of an envelope or napkin, for example, and use it as the starting point for the design.

Circuit Mind takes a different approach, utilising PI to read component datasheets and extract the necessary information to drive its deterministic design algorithms. The folks at Circuit Mind recently added the ability to automatically generate the power supply portion of the system.

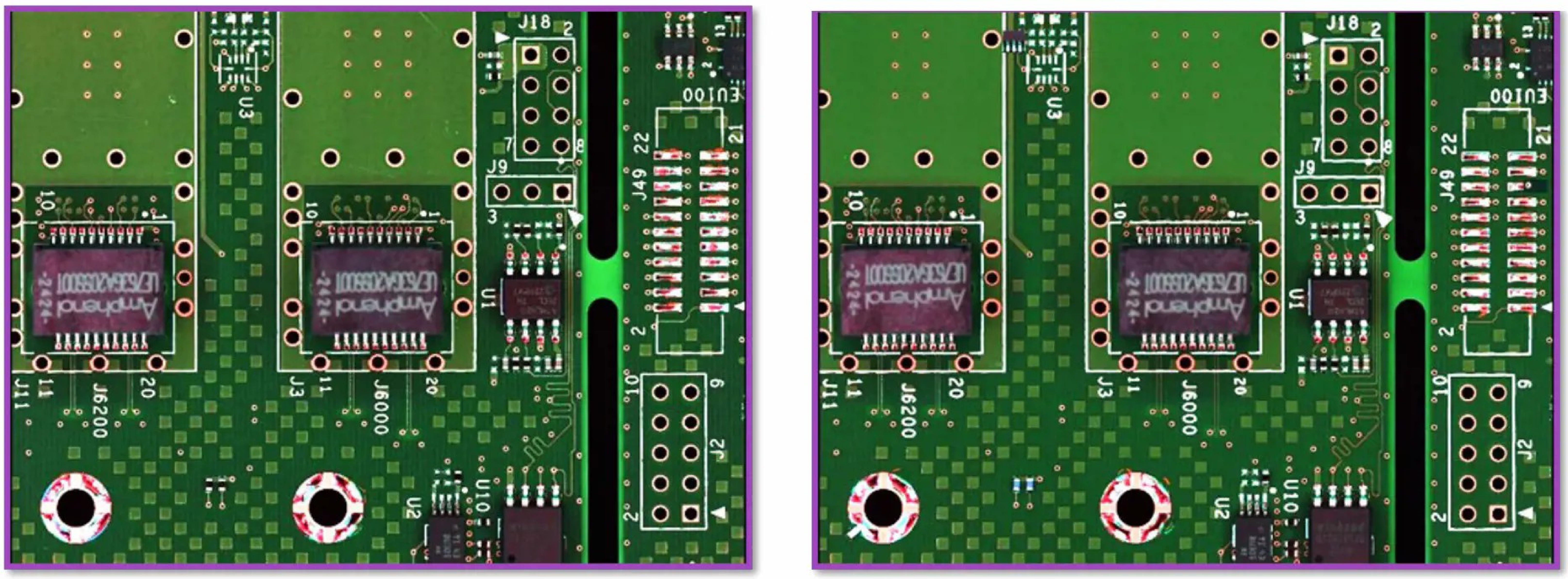
PI-enabled PCB layout tools are also starting to make their presence felt. Three companies of interest in this arena are AutoPCB (**autopcb.app**), JITX (**jitx.com**) and Quilter (**quilter.ai**). I hope to speak with the founders of all these entities in the next few weeks, and I'll report further in a future column.

PI-powered PCBA verification

I was just chatting with Eyal Weiss, a man who played a crucial role in developing a new technology. Eyal ultimately secured a billion-dollar deal. He was, let's say, 'sad' when thousands of circuit boards started to fail in the field. It turned out that the contract manufacturer had inadvertently used 10-year-old counterfeit capacitors, so all was nearly lost for the sake of a couple of cents' worth of component.

Vowing that this would never happen to him again, Eyal founded Cybord (**cybord.ai**). Its claim to fame is that, when installed in PCB assembly facilities, it can detect old components, counterfeit components, substituted components, added components and subtracted components, among many other anomalies. I don't know about you, but I prefer my PCBs anomaly-free!

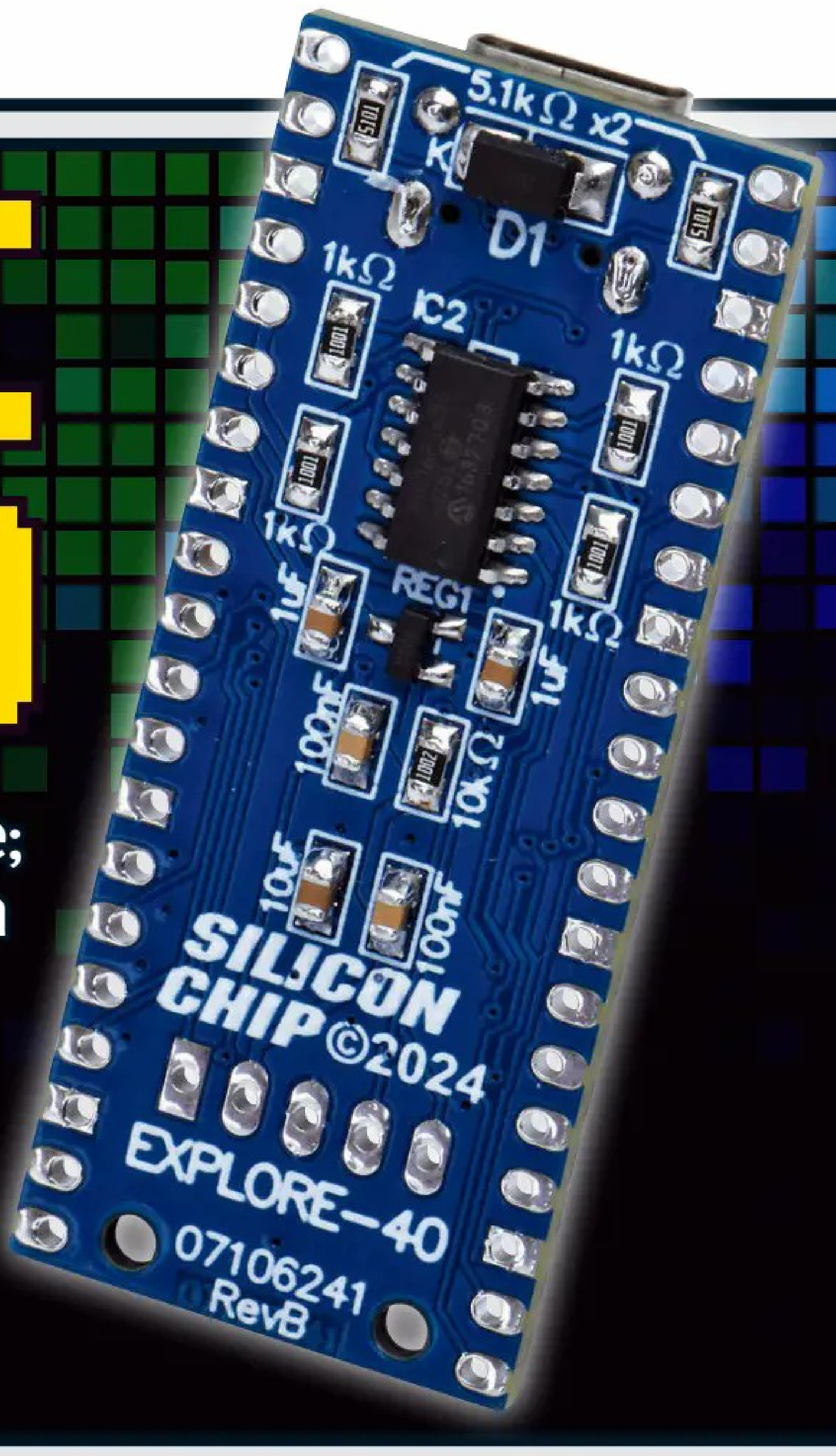
The exciting thing is that everything discussed in this column is just a tempting teaser for what is to come in the not-so-distant future!



Spot the difference: can you tell what differences Cybord found between the image of the reference board (left) and the later production variant (right)? Turn to page **43** for the answers.

MICROMITE EXPLORE-40

A wealth of software has been written for the Micromite; The Back Shed online forum is a great place to find much of it. This compact Explore-40 board is a Micromite in the same form factor as the popular Pico boards, allowing a Micromite to be used with hardware designed for the Pico.



PROJECT BY TIM BLYTHMAN

THE RASPBERRY PI PICO has taken a well-deserved place as one of the most popular microcontroller boards. It is cheap, easy to use and can be programmed in C, BASIC, MicroPython and even with the Arduino IDE.

Our Pico BackPack project in the March 2023 issue capitalised on those features, providing stereo audio and a microSD card interface with the 3.5in LCD panel that we had previously used with the Micromite V3 BackPack.

These new features can now be accessed from Micromite BASIC, since the Explore-40 board allows a Micromite processor to be plugged into the Pico BackPack. Thanks in part to the ongoing work of The Back Shed forum members, software is available to use these new features.

The Micromite Explore-40 is not just a Micromite/PIC32 breakout board. It has been designed to include niceties like an inbuilt USB-serial converter, plus some LEDs and pushbuttons.

The Explore-40

The Explore-40 is typical of minimal Micromite implementations that include the Microbridge USB-serial converter. The circuit resembles earlier Micromite boards like the Explore-28 from the September 2020 issue.

Since this board is patterned after the Raspberry Pi Pico and thus a bit larger than the Explore-28, we have taken the opportunity to add some extra features.

Circuit details

Fig.1 shows the circuit of the Explore-40. IC1 is a PIC32MX170F256B in a relatively large 28-pin SOIC package. This is the familiar 28-pin part we have used for many Micromite projects.

Its I/O pins are connected to pins on the pair of 20-way headers that match the pinout of the Pico. We'll explain our choices for this specific mapping a bit later. As the Pico has more pins than the 28-pin PIC32, there are some empty positions on those 20-way headers.

IC2 is a PIC16F1455 programmed with the Microbridge firmware. The Microbridge was originally published as a separate board in the May 2018 issue. It has since been incorporated into many Micromite designs. It can function as a USB-serial converter, allowing communication between a computer and the Micromite chip.

The Microbridge can also act as a programmer, allowing new firmware (such as a new version of Micromite BASIC) to be easily installed on the Micromite chip.

As such, it connects to the data lines on USB-C connector CON1, as well as the serial and programming pins of IC1. IC2 also drives LED1, which indicates its mode (USB-serial or programming) and shows serial traffic. Onboard pushbutton S2 selects IC2's mode.

Micromite Explore-40 Features & Specifications

- Allows a PIC32 Micromite processor to be plugged into a Pico socket
- All 28-pin Micromite I/O pins are available
- Onboard Microbridge serial interface/programmer
- USB-C socket for power and data
- Micromite BASIC software examples for all Pico BackPack features
- Supports LCD touch panel with backlight control
- Supports IR receiver
- Stereo audio output
- microSD card interface
- Realtime clock interface
- Add-on 3.5mm board provides 3.5mm stereo audio socket with Pico BackPack
- Power and status LEDs
- Reset and Mode pushbuttons
- In-circuit serial programming (ICSP) header for the PIC32 Micromite chip

CON1, the USB-C socket, has connections to power via the VBUS pins. The CC1 and CC2 pins are connected to ground via 5.1kΩ resistors, signalling to the USB source (eg, a computer) that it should supply 5V on the VBUS pins.

The VBUS voltage goes via schottky diode D1 to REG1, an MCP1700 3.3V low-dropout regulator. The diode also connects to pins 40 and 39 of the Pico headers, emulating that handy feature of the Pico boards. It means that an alternative source of 5V power can be fed into pin 39 (possibly via another diode) without any risk of back-feeding the USB power supply.

REG1 and its capacitors provide a 3.3V rail that powers IC1, IC2 and power indicator LED2 (the latter via a 1kΩ resistor). The 3.3V output is also available on pins 35 and 36 of the headers, as it is on the Pico boards. IC1's pin 1 (the $\overline{\text{MCLR}}$ reset input) has also been taken to pin 30 (RUN on the Pico).

Pins 3, 8, 13, 18, 23, 28, 33 and 38 of the headers are connected to ground, like the Pico, and we have connected as many of the PIC32's I/O pins as we can to the remaining I/O pins on the headers. Because of this compatibility, we're sure readers will find the Explore-40 handy in other situations where a Pico might be used.

IC1's $\overline{\text{MCLR}}$ reset pin, 3.3V, ground and the two ICSP programming pins are also available at the CON2 ICSP header, allowing the chip to be programmed by an external programmer. IC1 can also be reset by pressing S1, which pulls $\overline{\text{MCLR}}$ to ground. A 10kΩ resistor pulls this up otherwise. A reset button is one feature that the real Pico lacks!

Pin mapping

The mapping of the 40-pin header has been mostly chosen to match the functions of the Pico BackPack to that of the Micromite. For example, the Micromite has fixed SPI and I²C pins, so the mapping matches the wiring of these two peripherals on the Pico BackPack.

Similarly, the pins for interfacing with the LCD on the V3 BackPack have been arranged identically on the Explore-40. This allows identical Micromite OPTIONs to be used. The infrared/IR pin (Micromite pin

Micromite Explore-40 Kit from Silicon Chip (SC6991, ~£21)

A complete kit is available for the Micromite Explore-40 with all the parts listed in the parts list on page 11 (not including the Audio Breakout Board or Pico BackPack) – see page 78 for details.

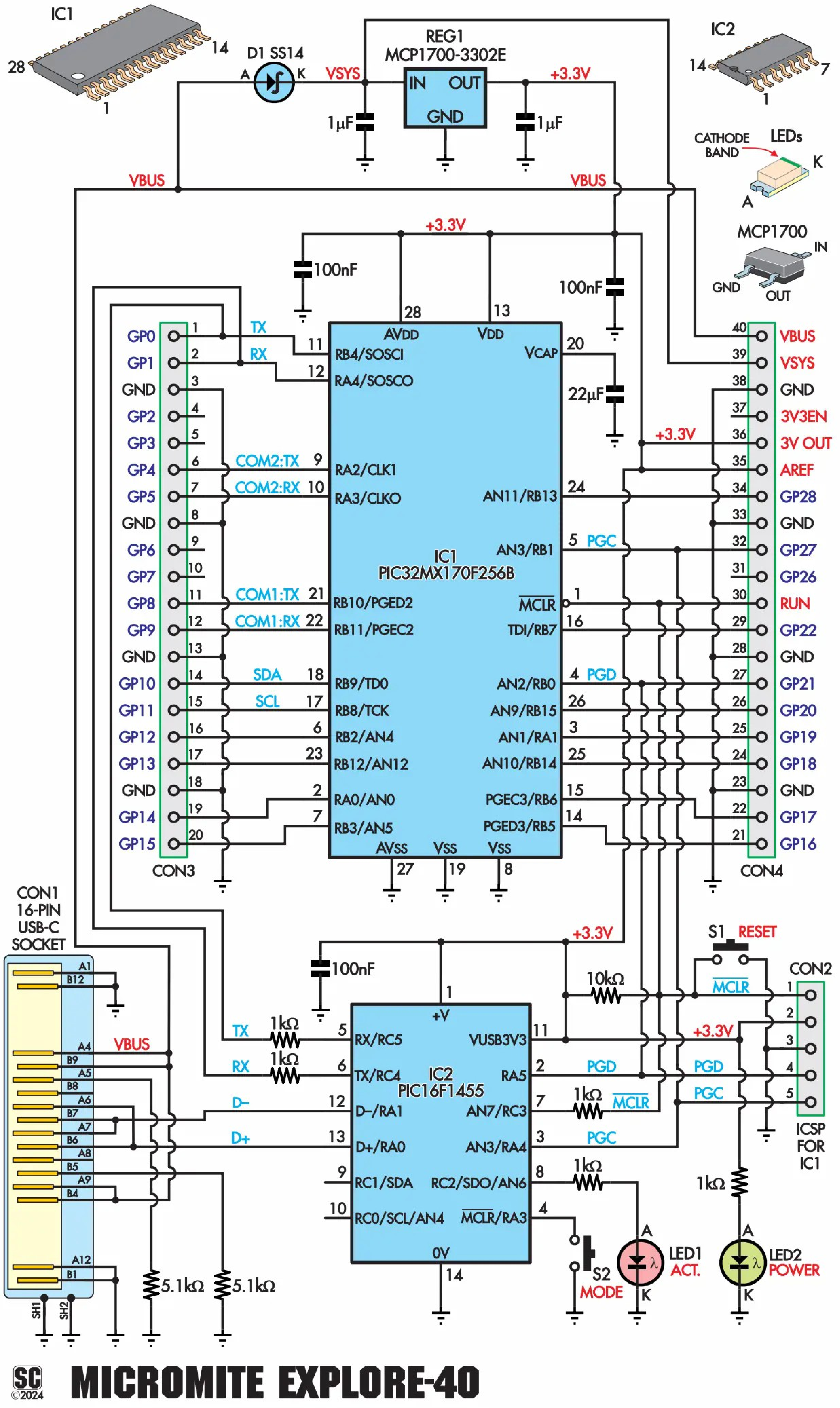

IC1
28
14
1
D1 SS14
A
K
VSYS
REG1
MCP1700-3302E
IN
OUT
GND
+3.3V
1μF
1μF
IC2
14
7
1
CATHODE BAND
LEDs
K
A
VBUS
VBUS
+3.3V
100nF
100nF
MCP1700
IN
GND
OUT
28
13
AVDD
VDD
TX
RX
11
12
RB4/SOSCI
RB4/SOSCO
VCAP
20
22μF
+3.3V
COM2:TX
COM2:RX
9
10
RA2/CLK1
RA3/CLKO
AN11/RB13
24
5 PGC
AN3/RB1
IC1
PIC32MX170F256B
COM1:TX 21
COM1:RX 22
RB10/PGED2
RB11/PGEC2
MCLR
TDI/RB7
1
16
SDA 18
SCL 17
RB9/TDO
RB8/TCK
RB2/AN4
RB12/AN12
RA0/AN0
RB3/AN5
4 PGD
AN2/RB0
AN9/RB15
AN1/RA1
AN10/RB14
PGEC3/RB6
PGED3/RB5
26
3
25
15
14
6
23
2
7
AVSS
VSS
VSS
27
19
8
GP0
GP1
GND
GP2
GP3
GP4
GP5
GND
GP6
GP7
GP8
GP9
GND
GP10
GP11
GP12
GP13
GND
GP14
GP15
CON3
VBUS
VSYS
GND
3V3EN
3V OUT
AREF
GP28
GND
GP27
GP26
RUN
GP22
GND
GP21
GP20
GP19
GP18
GND
GP17
GP16
CON4
CON1
16-PIN
USB-C
SOCKET
A1
B12
A4
B9
A5
B8
A6
B7
A7
B6
A8
B5
A9
B4
A12
B1
SH1
SH2
VBUS
+3.3V
100nF
S1 RESET
10kΩ
MCLR
CON2
TX
RX
1kΩ
1kΩ
5
6
12
13
9
10
D–
D+
+V
RX/RC5
TX/RC4
D–/RA1
D+/RA0
RC1/SDA
RC0/SCL/AN4
IC2
PIC16F1455
VUSB3V3
RA5
AN7/RC3
AN3/RA4
RC2/SDO/AN6
MCLR/RA3
0V
14
11
2
7
3
8
4
PGD
1kΩ
MCLR
PGC
1kΩ
+3.3V
PGD
PGC
1
2
3
4
5
ICSP FOR IC1
1kΩ
A
A
LED1 ACT.
LED2 POWER
S2 MODE
K
K
5.1kΩ
5.1kΩ
SC ©2024
MICROMITE EXPLORE-40

Fig.1: the Explore-40 has much in common with the Micromite V2 BackPack and the Explore-28, although we've added a USB-C socket, power indicator LED and a reset button. The I/O pin mapping to the two 20-pin headers is designed to allow the Micromite processor to work with the Pico BackPack and retain some compatibility with software designed for the V3 BackPack.

16) has also been connected to the IR receiver on the Pico BackPack.

Pins 21 and 22 on the Micromite have been connected to pins 11 and 12 of the Pico header; these are used for audio on the Pico BackPack and are a convenient pair for this purpose. If not used for audio, they can be used as the Micromite's COM1 serial port.

The serial console pins have been allocated to pins 1 and 2, allowing the console to be connected to the Pico Digital Video Terminal. That doesn't leave many pins spare to be allocated. We have connected pins with analog functions where possible, although the Pico has fewer than the Micromite.

We'll detail the OPTIONs and pins that should be used with the Pico BackPack later, when we explain the software features in more detail.

A small add-on

While putting together this design, we realised adding a 3.5mm audio output jack socket to the Pico BackPack would be a nice touch. We initially omitted this from the Pico BackPack because the board is quite tight for space.

To solve this, we've designed a very small daughterboard that can be connected to the Pico BackPack, breaking out the CON3 audio connector into a 3.5mm stereo socket. It is shown in Fig.2.

A 3.5mm jack socket breakout board for the Pico BackPack

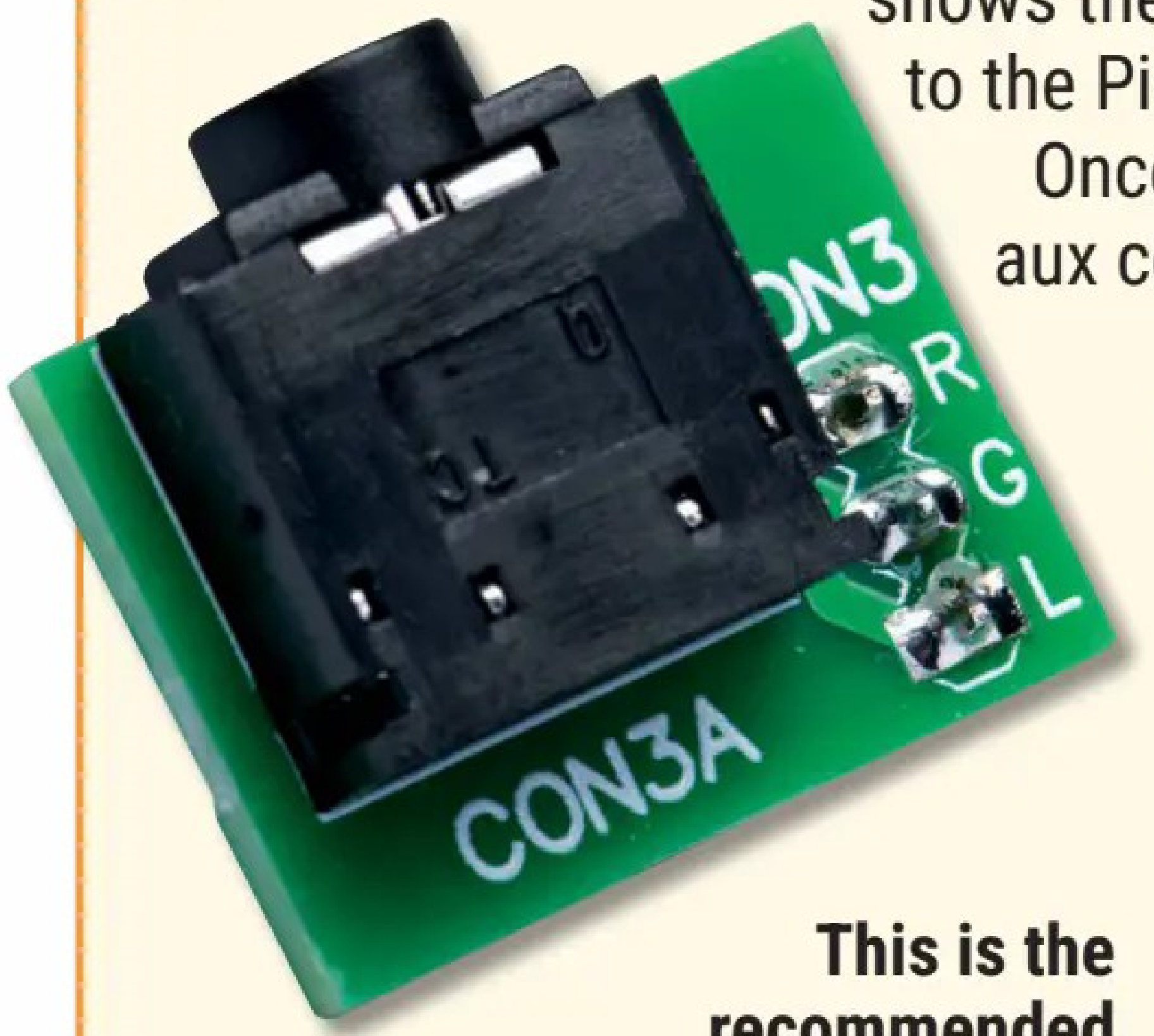
Building this board is simple, as you can see from our photos. As long as you connect the R, G and L pins to the matching pins on CON3 of the Pico BackPack, the board can be installed in a few different ways.

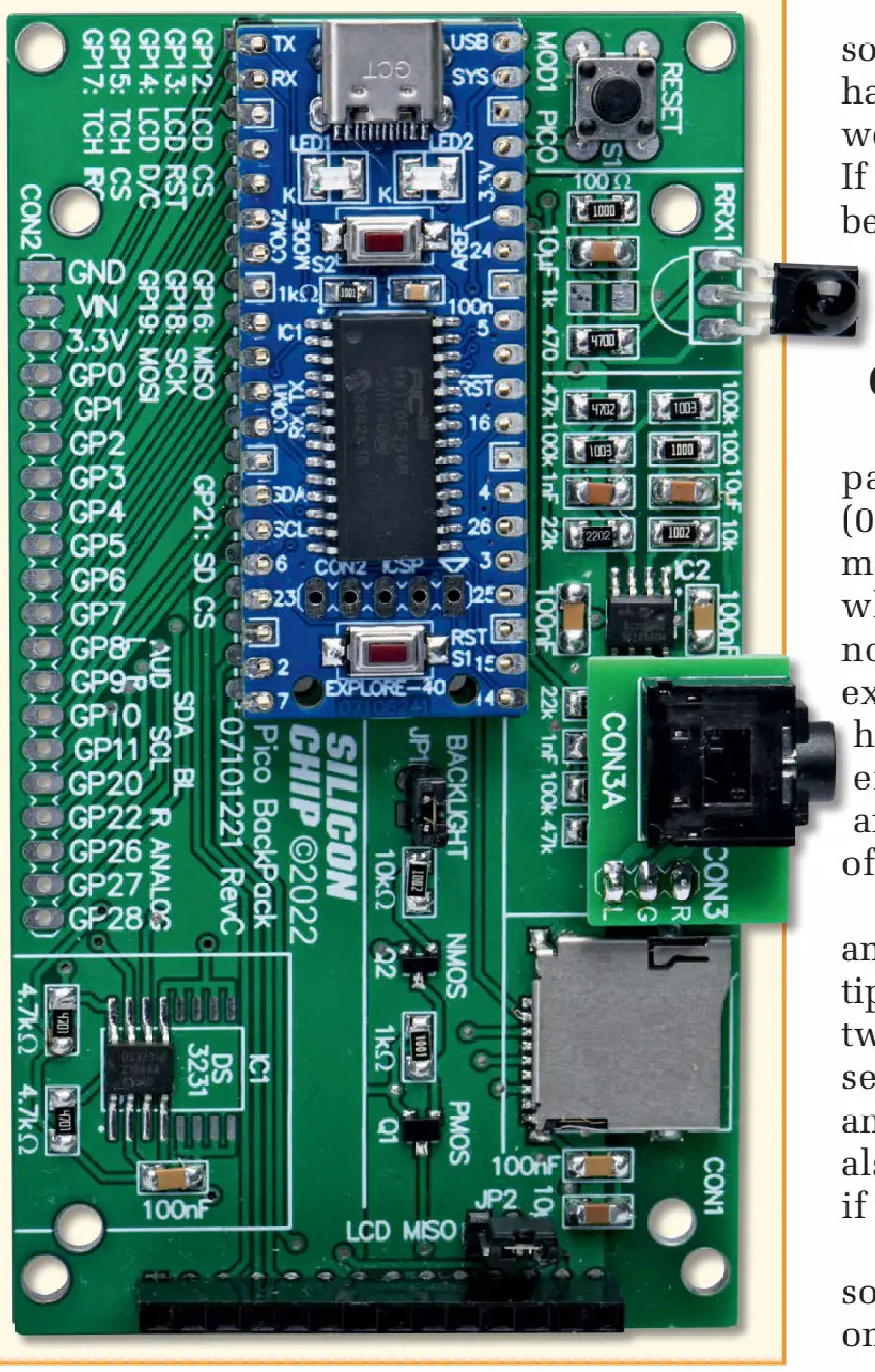
It can be mounted on either side of the board, giving four main configurations. We think the method shown in our photos is the simplest, gives a compact result and does not put the audio socket awkwardly close to the microSD card socket.

The Audio Breakout extends slightly beyond the Pico BackPack and is intended to sit just inside a UB3 Jiffy box so that the socket can be accessed through a small hole in the side.

We suggest fitting the audio socket to the PCB first. That will allow you to easily check that your chosen positioning does not foul any other components. The photo shows the assembly of the listed parts that can then be fitted to the Pico BackPack.

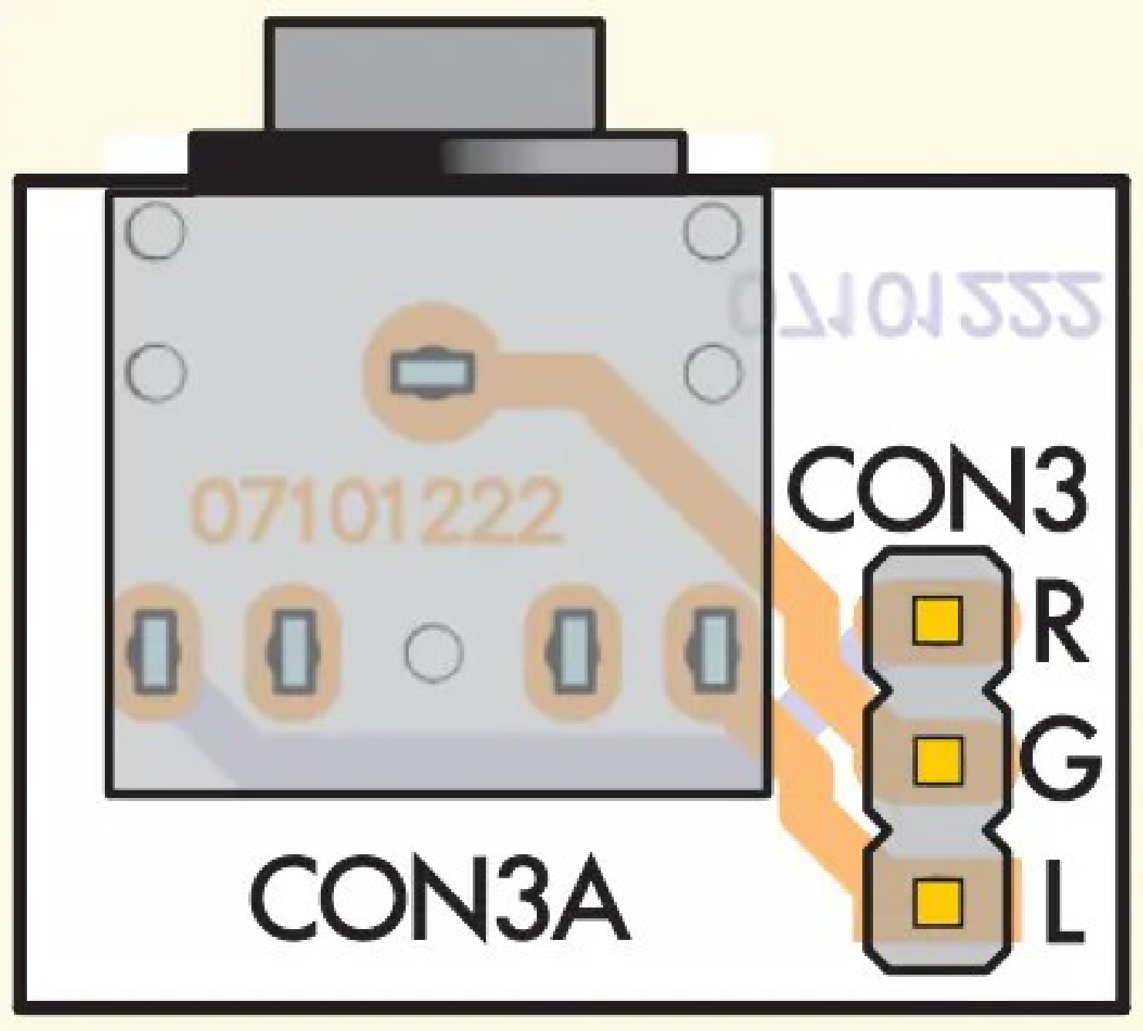
Once fitted, you can simply plug in headphones or an aux cord to hear audio from CON3 on the Pico BackPack.





This is the recommended placement of the 3.5mm jack socket breakout board on the Pico BackPack, sitting above some passive components in the audio section. Although it's designed to work with the Pico BackPack, you can also use it for breadboarding or prototyping.

Fig.2 (below): when assembling the breakout board, ensure the socket is pushed firmly against the PCB. We used straight headers, but you could use right-angled headers.



You don't need to use the Explore-40 to use the daughterboard; it can also be used with a Pico or Pico W. You can see it in our photos, mounted above the Pico BackPack PCB. We have a panel showing how to build this board and add it to the Pico BackPack.

Programming the chips

IC1 can easily be programmed via IC2 once you have built the board, but IC2 is best programmed before it is soldered to the board, especially as there is no ICSP header for it.

It is possible to use a Micromite to program a Microbridge; there are notes on how to do that included with the Microbridge firmware at https://pemag.au/Shop/6/4269

Still, it is easier to program IC2 with something like a PICkit or SNAP if you have an appropriate SMD adaptor, so we recommend doing that if possible. If you buy a kit from us, both ICs will be programmed already; there is also the option to buy programmed chips separately.

Construction

The Explore-40 uses mainly SMD parts, including SOIC ICs, M2012 (0805 imperial) passive components measuring 2.0 × 1.2mm, and a somewhat fine-pitch USB-C socket. It is not super difficult, but neither is it extremely easy; it would be ideal to have some SMD soldering experience before assembling it. There are also components on both sides of the PCB.

You will need the usual SMD tools and consumables. A fine- or medium-tipped soldering iron, solder, flux paste, tweezers and good ventilation are essential. Some solder-wicking braid and a means of securing the PCB are also advised. Blu-Tack will do the job if you don't have a PCB vice.

You should also have a suitable solvent for cleaning up flux, such as one recommended by your flux supplier. Alternatively, isopropyl alcohol or methylated spirits will be effective for most fluxes. Fig.3 shows the PCB overlays, which you should refer to during assembly.

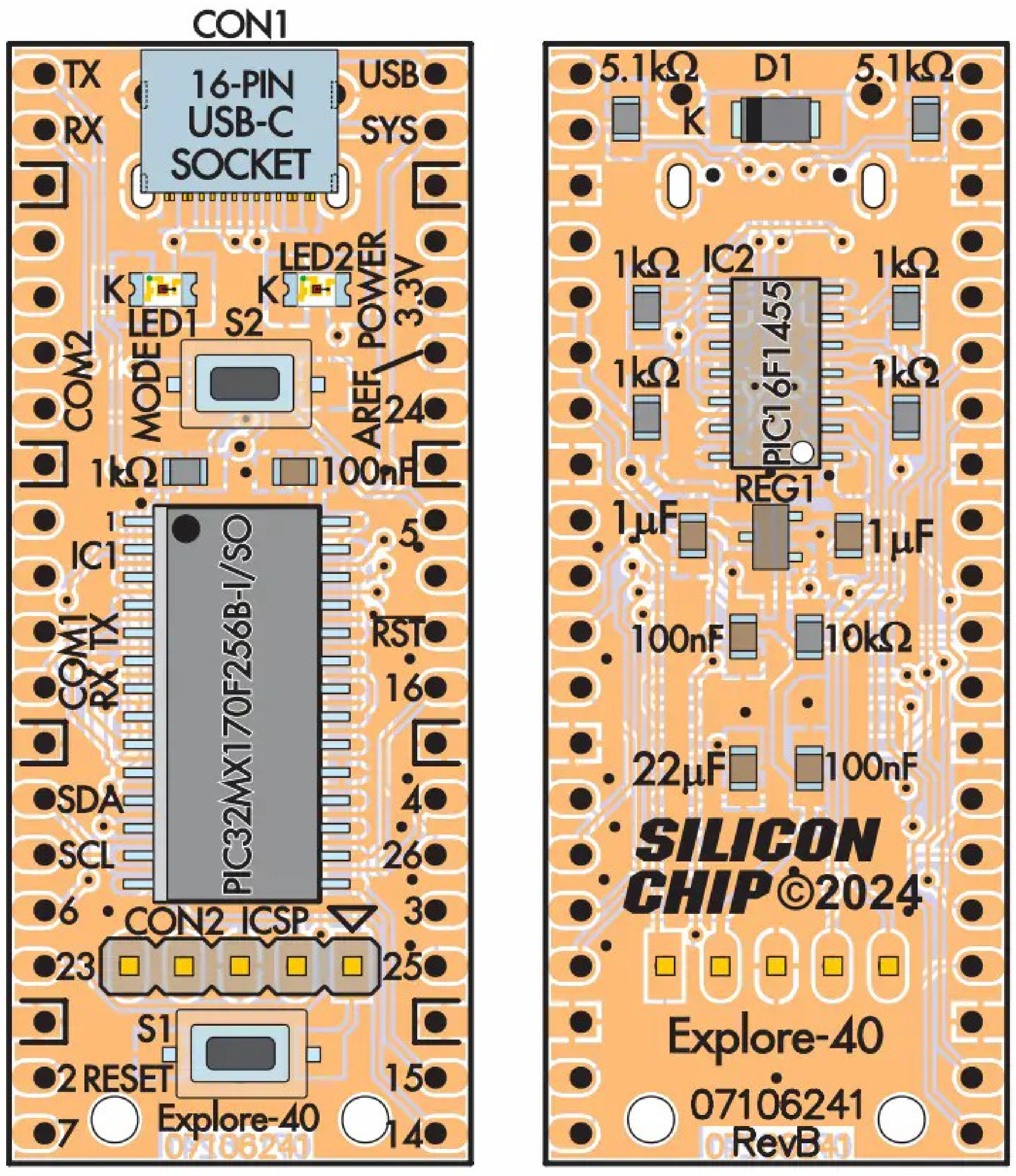


Fig.3: we've placed components on both sides of the PCB to best use the available space. The CON1 USB-C socket and the two microcontrollers have the tightest pin pitches, so they should be fitted first. Avoid using too much solder for CON1 through-hole mounting pins in case it bridges to D1 or the nearby resistors. This diagram is shown at 150% of actual size for clarity.

Start by soldering CON1, the USB-C socket, since it has the closest pin pitch. It will also be difficult to get to once other components are installed. Apply flux to the pads and slot the socket into its holes on the top of the PCB. Clean the iron's tip and add a small amount of fresh solder.

The end-most leads are a bit wider, so tack one of those in place, then check that the other leads are aligned to their pads and that the part is flat against the PCB. Adjust it until you are satisfied. The locating posts should help here.

You can then solder the mounting pins from the reverse of the PCB. It might help to add some flux to the bottom and top of those pins to help the solder take. Try not to add too much solder to the mounting pins, as it might get in the way later.

Next, solder the remaining pins of CON1 on the top of the PCB. Use the braid and extra flux to remove any bridges that have formed. Place the braid on the solder, apply the iron and gently move both away together once the solder has been taken up.

Fit the two ICs next, being sure to get the correct orientation. IC2 faces the opposite direction to IC1 and is on the opposite side of the PCB. Add flux to the PCB, rest the ICs in place and tack one lead before soldering the others. Adding flux to the pins before soldering will help it flow. Check for bridges after soldering and remove any with more flux paste and the solder wick.

Regulator REG1 mounts on the same side as IC2. It's easy enough to solder but small enough to lose sight of easily. Add some flux and place it as shown. Tack one lead, then check the alignment of the others before soldering.

The diode mounts on the opposite side of the board from the USB-C socket. Ensure that the PCB's cathode mark matches the diode orientation and avoid bridging its pads to the socket's mounting pins.

Now solder the remaining parts on the underside of the PCB methodically. The resistors will have small codes printed on top (per the parts list) but the capacitors will be unmarked. You may be able to tell them apart by their thickness if you manage to get them mixed up. In each case, add flux to the pads, rest the part in place, tack one lead, then check and solder the other.

Next come the two LEDs on the top side of the board. We recommend using red for LED1 (MODE) and green for LED2 (POWER), although you could choose your own scheme. You can test the colour and polarity of the LEDs with a multimeter set to diode mode.

The cathode will be the end connected to the black multimeter lead when the LED lights up. Fit the LEDs with the cathodes towards the COM2 silkscreen marking (the overlay also shows a K near each cathode).

Solder the last 1kΩ resistor and 100nF capacitor. Clean both sides of the PCB thoroughly with your chosen flux solvent and allow the PCB to dry. It's then a good time to inspect the soldering for any bridges or dry joints you might have missed. If you find any, fix them before proceeding.

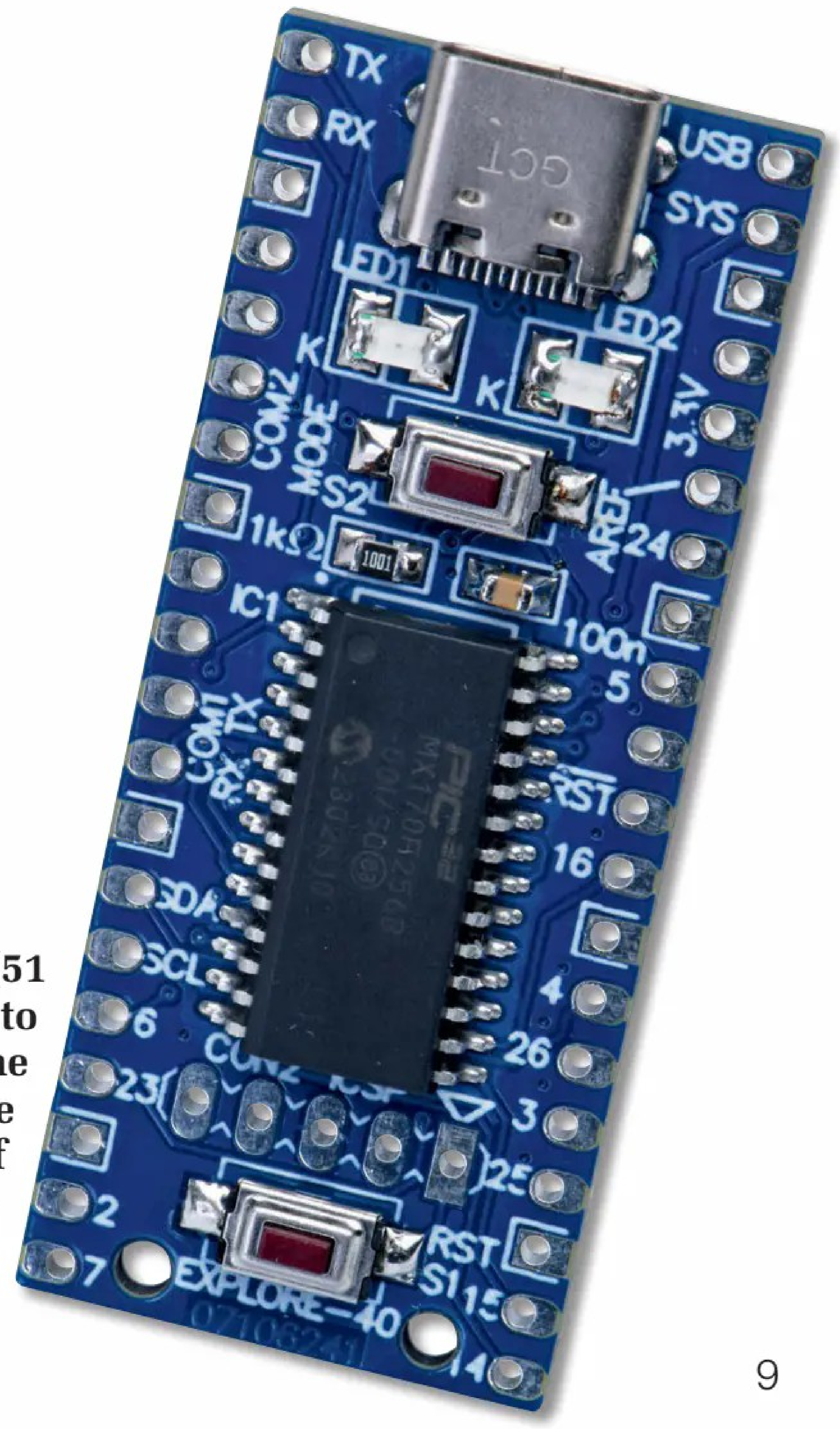
Fit the two tactile switches next. They have much larger pads, making them easier to solder than the other parts. Your board should look like the photos now.

Testing

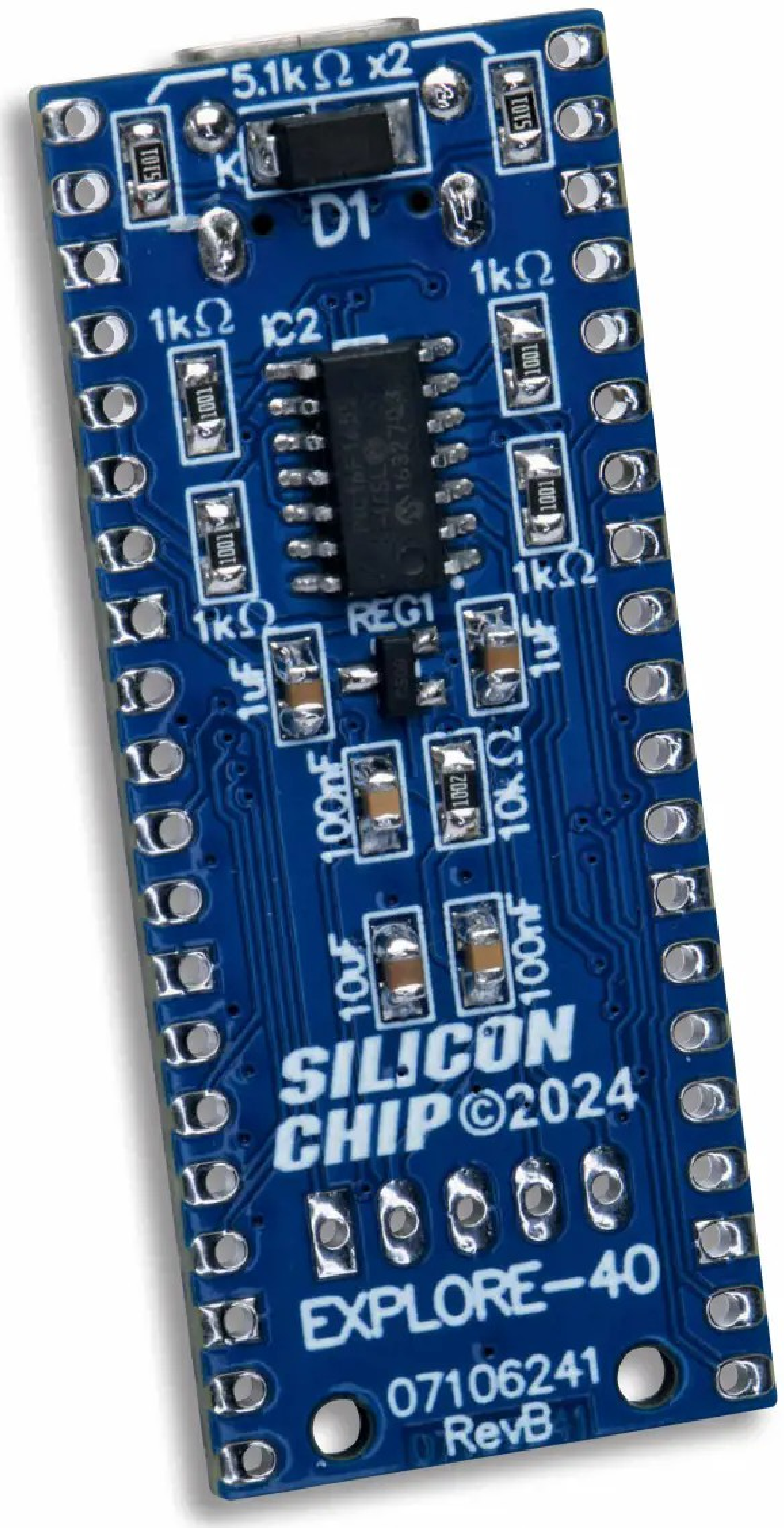
There are still some parts to fit, but now is a good time to do some initial tests. Connecting USB power to CON1 should cause LED2 to light up. The 3.3V pin should measure between 3.2V and 3.4V relative to ground. Pressing S2 should cause LED1 to light up, assuming IC2 is programmed correctly.

If something is not right, check for 5V at the USB pin, at upper right, and around 4.7V (due to the diode) at the SYS pin below it. Check the USB-C socket and 5.1kΩ resistors if the USB voltage is absent. An absence of voltage at the SYS pin suggests the diode is reversed or not connected, while a lack of 3.3V could point to a problem with the regulator or a short circuit on the 3.3V rail.

If you need to fit the CON2 ICSP header to program IC1, do that now. Be aware that you may not be able to leave CON2 attached afterwards since it might be too tall to fit between the Pico BackPack PCB and the LCD



The Explore-40 is a compact board (51 × 21mm) that allows the Micromite to substitute for a Raspberry Pi Pico in some circumstances. IC1 and the two LEDs are the polarised components on the top of the PCB. We recommend using red for LED1 and green for LED2.



The underside of the Explore-40 shown at actual size; note the orientations of IC2 and D1. REG1 is also polarised, but its correct orientation should be obvious.

panel. You can use IC2 to program IC1, after all.

If you connect the Explore-40 to a computer and open a serial terminal program such as TeraTerm, you should be able to communicate with the Micromite firmware. The default baud rate is 38,400. You can press S1 and check that the Micromite's boot message is printed via the terminal.

The Explore-40 is now complete enough to plan how you will fit it to the Pico BackPack. The most significant difference is that the Explore-40 has components on its underside, so it will not mount flush like a Pico could.

The following assumes that you are fitting the Explore-40 to a Pico BackPack with an LCD panel above. If you just plan to use it on a breadboard, for example, you just need to be sure that the pins align with the sockets in the breadboard.

Fitting it to a Pico BackPack

For our prototype, we used low-profile header sockets and removed the plastic shroud from the pin headers to allow the board to be swapped (eg, for a Pico) if needed. However, we found that quite fiddly to achieve.

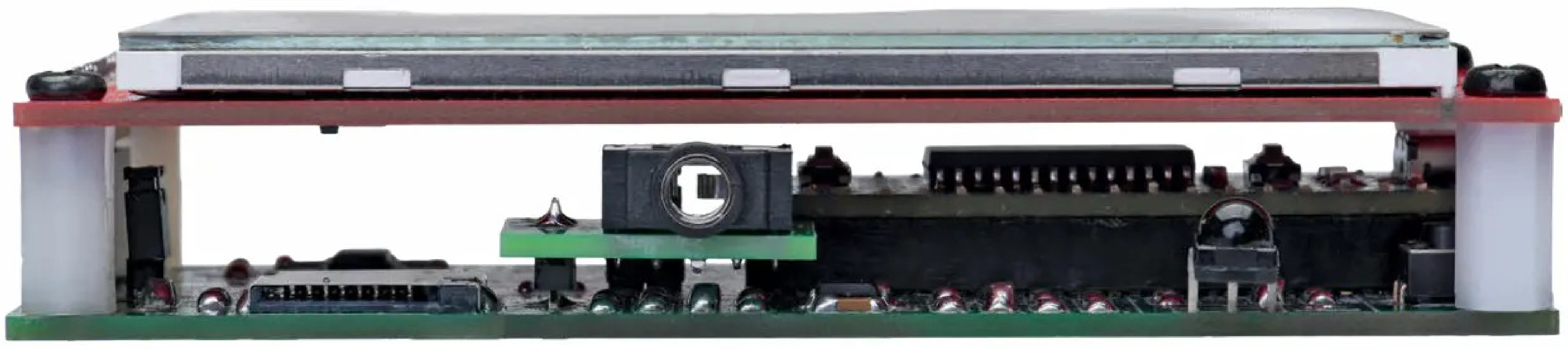
As you can see in the photo below, there is very little clearance above the Explore-40, even though we removed the SD card socket from the LCD panel above. Still, that is an option to consider since there is no connection to the SD socket on the LCD panel from the Pico BackPack.

If you want to do that, use a pair of flush nippers to gently cut and detach each pin from the SD card socket, then use a soldering iron to remove the remnants of each pin. Follow with some solder-wicking braid and flux paste to remove any solder residue.

If you are happy to permanently solder the Explore-40 to the Pico BackPack, the height of the plastic spacers on standard pin headers will prevent the underside components from touching the PCB below. To do this, sandwich the headers between the Pico BackPack and Explore-40 PCBs, then tack a few pins in place before soldering the remainder and trimming the excess lengths away.

If you are doing something different, we recommend test-fitting the parts first to be sure they will fit and not cause any fouling with the LCD panel above. It's also possible to fit the Explore-40 to the underside of the Pico BackPack PCB, although that will make it difficult to access the buttons or see the LEDs.

If you want to do that, we suggest using socket headers fitted to the top of the Explore-40 that will mate with pin headers mounted on the underside of the Pico BackPack.



We used low-profile header sockets to mount our prototype Explore-40, but if you solder it directly to the BackPack PCB using standard header pins, you will gain clearance since the Explore-40 will sit lower. With some care, the unused SD card socket on the underside of the 3.5in LCD panels can be removed, giving extra clearance below. Use solder-wicking braid to clean off any excess solder left behind.

Software support

Combining the Explore-40 with the Pico BackPack (and 3.5in LCD panel) brings two new features that were not present on the V3 Micromite LCD BackPack. These are the microSD card and audio output.

First we'll recap the features that are shared with the Micromite V3 BackPack and how they are configured and used. This will be a quick way to check that the Explore-40 is working as expected.

These features should all behave identically to a Micromite V3 BackPack. Note, though, that the Explore-40 and Pico BackPack lack the RAM or FLASH IC and temperature sensors that the V3 BackPack includes.

The Micromite firmware does not have a built-in driver for the 3.5in LCDs, but there is a loadable driver developed by Peter Mather. We have customised this to suit the configuration of the Explore-40 and Pico BackPack hardware; it is the "3.5IN DRIVER.BAS" file in the software downloads package.

The code is much the same as that found in the Display Drivers folder of the Micromite firmware download. We have just changed the line in the MM.STARTUP subroutine to suit our pin allocation. The "3" at the end indicates a landscape configuration, with the microSD card socket near the top of the screen.

Load this file onto the Micromite (for example, using the AUTOSAVE command), then perform a LIBRARY SAVE and restart the Micromite by pressing S1 or entering the CPU RESTART command. You should see the screen clear and you can run the GUI TEST LCDPANEL command to confirm it is working.

To configure, calibrate and test the touch panel, use these commands:

```
OPTION TOUCH 7,15
GUI CALIBRATE
GUI TEST TOUCH
```

If the required components and jumpers are fitted to the Pico BackPack, the backlight is also driven from IC1's pin 26, just like the V3 BackPack. This can be controlled using PWM channel 2A. The following will set the duty cycle and backlight brightness to 50%:

```
PWM 2,250,50
```

IR receiver & realtime clock

The IR receiver on the Pico BackPack is routed to the dedicated Micromite IR pin, pin 16, so the IR receiver can be used by simply setting up the IR interrupt with the IR command. The command would be something like:

```
IR DevCode, KeyCode, IR_Int
```

A basic interrupt subroutine to test this could be:

```
SUB IR_Int
  PRINT “DEVICE:” DevCode
  “KEY:” KeyCode
END SUB
```

The RTC commands support the realtime clock chip:

```
RTC GETTIME
RTC SETTIME year, month, day,
  hour, minute, second
```

You can then retrieve the current time and date from the TIME$ and DATE$ variables.

MicroSD card support

The Micromite lacks a native driver for interacting with SD cards. Peter Mather has again done some excellent work in creating a CSUB driver to do that. However, there are a few provisos to using this software. Since the Micromite does not have an interface for file handling (unlike the Micromite Plus), everything is done via calls to the CSUB.

The driver is quite simple and cannot do things like create or append to files. So, if you wish to write to a file, the recommendation is to create a large file on the card, which the driver can then overwrite. Even with these restrictions, the driver takes up about one-sixth of the flash memory available for programs. More background information on this and suitable code can be found at https://pemag.au/link/abxr

We have configured pin 4 as the $\overline{\text{CS}}$ (chip select) pin for the microSD card socket. This is the same pin that is wired to the SD card socket on the LCD panel for the Micromite V3 BackPack. So you could try this on a V3 BackPack, although we haven’t tested it.

Parts List – Micromite Explore-40

1 51 × 21mm double-sided PCB coded 07106241
1 16-pin USB-C data and power socket (CON1) **[GCT USB4105]**
1 5-way pin header, 2.54mm pitch (CON2; **optional**, for ICSP)
2 20-way pin headers, 2.54mm pitch
2 SMD 2-pin tactile switches (S1, S2)

Semiconductors
1 SS14 40V 1A schottky diode, DO-214AC/SMA (D1)
1 PIC32MX170F256B-50I/SO 32-bit microcontroller programmed with the Micromite firmware, wide SOIC-28 (IC1)
1 PIC16F1455-I/SL 8-bit microcontroller programmed with the Microbridge firmware, SOIC-14 (IC2)
1 MCP1700-3.3 3.3V low-dropout voltage regulator, SOT-23 (REG1)
1 red M3216/1206/SMA SMD LED (LED1)
1 green M3216/1206/SMA SMD LED (LED2)

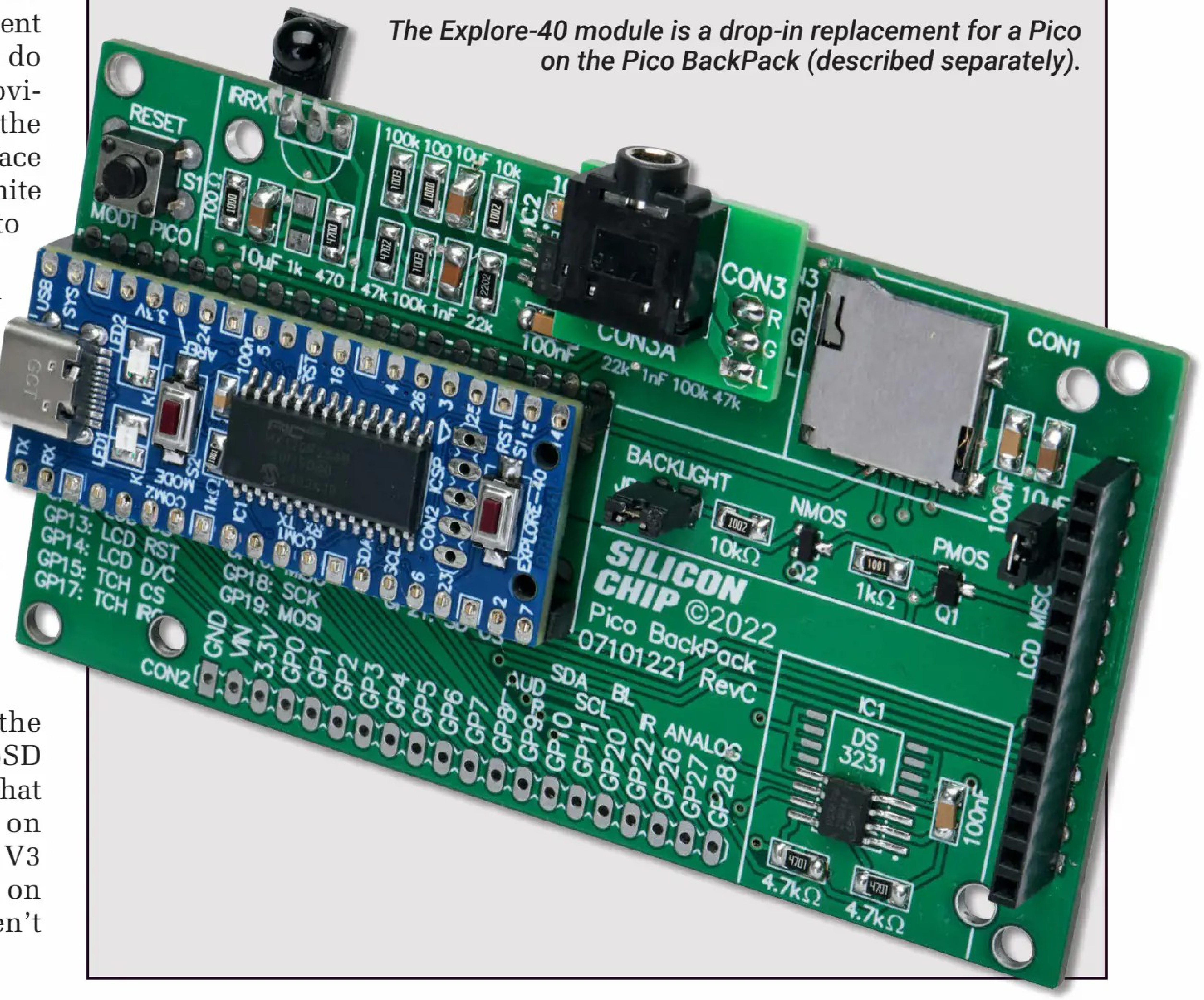
Capacitors *(all SMD M2012/0805, X7R)*
1 22µF 10V X5R/X7R
2 1µF 16V
3 100nF 50V

Resistors *(all SMD M2012/0805, ⅛W)*
1 10kΩ (code 1002 or 103)
2 5.1kΩ (code 5101 or 512)
5 1kΩ (code 1001 or 102)

Optional extras
1 Pico BackPack (without Raspberry Pi Pico) plus 3.5in LCD (March 2022)
1 3.5mm jack socket breakout board (see panel and parts below)

Audio Breakout Board

1 double-sided PCB coded 07101222, 20 × 15mm
1 stereo 3.5mm PCB-mounting jack socket (CON3A) **[PJ-307 or similar]**
1 3-way pin header (CON3)



The Explore-40 module is a drop-in replacement for a Pico on the Pico BackPack (described separately).

The driver file is named “SDCARD_SPI1.BAS”. It is installed similarly to the LCD panel driver, using the AUTOSAVE and LIBRARY SAVE commands.

We’ve also created a HEX file that contains these two libraries loaded into a working copy of Micromite BASIC version 5.05.05, named “MM BASIC SD ILI9488.HEX”. You can load this with the onboard Microbridge or a PICkit programmer.

Audio support

The audio driver is another CFUNCTION that is controlled via calls with various parameters. This is based on a similar driver we created for the Advanced GPS Computer in the June & July 2022 issues.

This uses a pulse-width modulation (PWM) output to synthesise an analog voltage signal, with the PWM switching frequency being filtered out by a low-pass filter attached to that pin. The analog voltage is varied using a timer interrupt to update the PWM duty cycle for each sample to be played.

The big difference is that this driver is capable of stereo output, although it is limited to eight bits of resolution and an 8kHz sampling rate. Given that the Micromite has enough flash memory to play only seven seconds of audio, or enough RAM for about six seconds, we think it is a fair compromise.

The AUDIO folder in the software downloads contains several files, including the CFUNCTION driver, some encoded audio samples and BASIC code to demonstrate how to use the driver.

The samples are created as CFUNCTIONs, although they do not contain executable code. They consist of a 32-bit header that indicates how many bytes are in the sample, followed by that many bytes. Stereo samples are stored with the left channel data first. A mono sample played in stereo mode will play twice as fast since two bytes are used every sample period.

The driver is installed by loading the “CFUN_LIBS.BAS” file onto the Micromite, then using the LIBRARY SAVE command. Since the CFUNCTION returns a value, we need to do something with that value, like print it. Use this to start the driver:

```
PRINT AUDIO(0)
```

A sample is used by loading its BASIC file, then performing a LIBRARY SAVE. Tell the driver where the sample is located like this:

```
PRINT AUDIO(1,
  PEEK(CFUNADDR SAMPLE_NAME))
```

Then start playback with:

```
PRINT AUDIO(2)  'mono
PRINT AUDIO(12) 'stereo
```

The sound will play in the background and stop automatically. Using values 6 (mono) or 13 (stereo) as parameters will cause the playback to loop endlessly. Playback can be forced to stop with:

```
PRINT AUDIO(3)
```

You can also wait for playback to finish with:

```
DO WHILE AUDIO(4)<>0:LOOP
```

The “BASIC_SUBS.BAS” file has some more sample code and variables that can be used to make it easier to see what each parameter does.

The file named “AUDIO MMBASIC.HEX” contains the libraries, samples and BASIC code, alongside a working copy of Micromite BASIC version 5.05.05.

Notes

The 28-pin Micromite has somewhat limited peripherals, so there are some limitations. For example, the timer that provides the interrupt to fetch new audio samples is the same one used for the IR decoder. So we don’t think it is possible to use the IR and audio features at the same time, although it should be possible to switch between them.

The audio output uses two of the remappable PWM channels, so the PWM feature on pins 4 and 5 cannot be used at the same time as the audio. Pin 4 is mapped to the microSD card socket, so we expect it will be used for that feature instead.

In any case, the CFUNCTION libraries take up quite a bit of program memory, as do audio samples, if kept in flash memory.

Note also that the PIC32MX170F256B microcontroller can be programmed in the low(er)-level C language using the MPLAB X IDE.

For example, we have also seen an MPLAB X project that can play stereo audio from an SD card, so this could be another way to program this hardware combination.

Summary

If you are a Micromite fan and yearning for the features of the Pico BackPack, the Explore-40 is the perfect way to bridge that gap. It adds microSD card support and stereo audio features that were not included on the earlier Micromite BackPack designs.

There are some limitations to what the Micromite can achieve, but it is still a handy platform for learning the BASIC language.

The Explore-40 also adds nice touches, like the modern USB-C socket and reset button. These features can be handy regardless of whether the Explore-40 is used by itself, on a breadboard or as part of a BackPack.

MORE MAGAZINES: XSAVA.XYZ
CON1
CON3
J1
RX1
HS0003577A2

The 3.5mm jack socket breakout board is a neat fit under the LCD panel, even when mounted on header pins. Like the Explore-40, you should trim any excess pin length with flush nippers or sidecutters.

Subscribe to SILICON CHIP

SILICON CHIP
Single Sideband Shortwave Receiver
Hot Water System SOLAR DIVERTER
Solar-optimised hot water system heating using power purely from excess solar generation

Silicon Chip is one of the best DIY electronics magazines in the world. Each month is filled with a variety of projects that you can build yourself, along with features on a wide range of topics from in-depth electronics articles to general tech overviews.

	Print	Combined	Online
6 months	$100 ≈£50 ≈€58	$110 ≈£55 ≈€64	$50 ≈£25 ≈€29
1 year	$195 ≈£97.50 ≈€113	$215 ≈£112.50 ≈€125	$95 ≈£47.50 ≈€55
2 years	$380 ≈£190 ≈€220	$415 ≈£207.50 ≈€241	$185 ≈£97.50 ≈€107

- *Combined subscriptions include both the printed magazine and online access.*
- *Prices are in Australian dollars (AUD) unless otherwise specified.*
- *Prices listed are for everywhere in the world except Australia and New Zealand. View siliconchip.com.au/Shop/SubRates for a full list of current prices.*
- *Price estimates are just that and should only be used as a guide. We can only take payment in the form of Australian dollars.*

Try our Online Subscription – now with PDF downloads!

An online issue is perfect for those who don't want too much clutter around the house and is the same price worldwide. Issues can be viewed online, or downloaded as a PDF.

Published in Silicon Chip

SSB Shortwave Receiver; June-July 2025

Versatile Battery Checker; May 2025

RGB LED Analog Clock; May 2025

To start your subscription go to siliconchip.com.au/Shop/Subscribe

The MIPI I3C Bus

There is a brand-new serial bus on the block named I3C (Improved Inter-Integrated Circuit). It is beginning to appear in mainstream parts; here's what you need to know about it.

By Andrew Levido

Right now, if you want to connect a peripheral like a sensor or an EEPROM to a microcontroller, you would probably use either the SPI (Serial Peripheral Interface) or I^{2}C (Inter-Integrated Circuit) bus. These are tried and tested serial interfaces that date back to the 1980s.

The new I3C standard was developed by the MIPI Alliance (www.mipi.org) and incorporates key attributes of the venerable I^{2}C and SPI interfaces, as well as some interesting new features such as dynamic addressing, in-band interrupts, high data rate modes and hot-join capability. As the name suggests, I3C is closely aligned with I^{2}C. In fact, I^{2}C devices can even be used on an I3C bus.

Like SPI and I^{2}C, I3C uses a controller, typically a peripheral within a microcontroller, and one or more targets. Fig.1 shows a comparison between an I^{2}C, SPI and I3C bus, each connecting a controller (master) with three targets (slaves) that can each interrupt the controller. With I^{2}C, the targets are addressed via the bus, whilst SPI requires a separate chip select ($\overline{\text{CS}}$) signal for each target.

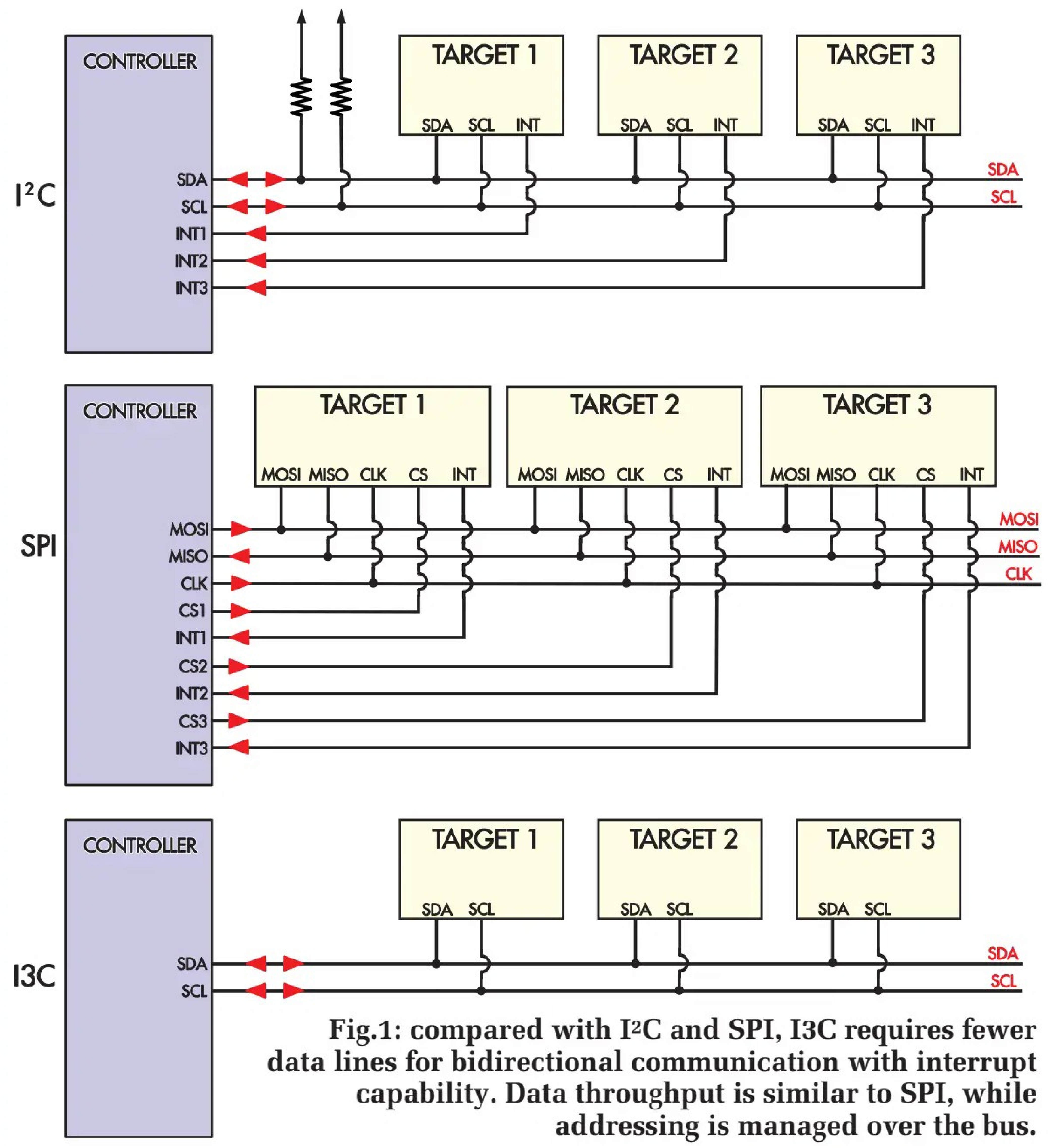


Fig.1: compared with I^{2}C and SPI, I3C requires fewer data lines for bidirectional communication with interrupt capability. Data throughput is similar to SPI, while addressing is managed over the bus.

I^{2}C uses bidirectional data transfer on the data line (SDA), while SPI requires two unidirectional lines (MISO and MOSI) for bidirectional communication. Both require separate interrupt lines if targets are to interrupt the processor asynchronously.

I3C uses device addressing and bidirectional data transfers, just like I^{2}C, but does not require dedicated interrupt signals, since targets can send interrupts to the controller via the bus.

The best of both worlds

The I^{2}C protocol uses open-drain drivers to achieve bidirectional signalling on the clock and data lines. High-to-low transitions are actively driven, but low-to-high transitions rely on pull-up resistors, so the signal rise time is limited by the resistor value and bus capacitance.

The maximum data rate that can be achieved over I^{2}C is therefore significantly lower than SPI, where the bus is actively driven high and low by push-pull drivers.

For I^{2}C, the maximum clock speed is typically 400kHz, or 1MHz with Fast mode+ drivers. There is a standard allowing speeds up to 3.4MHz, but it is not widely supported. The SPI interface does not have standard clock rates but can typically operate at a maximum clock speed of 10-20MHz.

I3C uses drivers that can operate in open-drain mode when required, but they switch to push-pull mode whenever possible to maximise the data transfer rate. External pull-ups are not required – these are provided by the controller when necessary.

Besides using open-drain drivers when communicating with legacy I^{2}C devices, open-drain drivers are used whenever bus arbitration is required; we'll cover that in more detail later.

Data transfers generally use the push-pull drivers.

As a result, the I3C bus operates at a range of clock speeds. In push-pull mode, the maximum clock speed is 12.5MHz, while in open-drain mode, it is 2.5MHz. The clock speed drops to 400kHz (or 1MHz for Fm+) when communicating with legacy I²C devices.

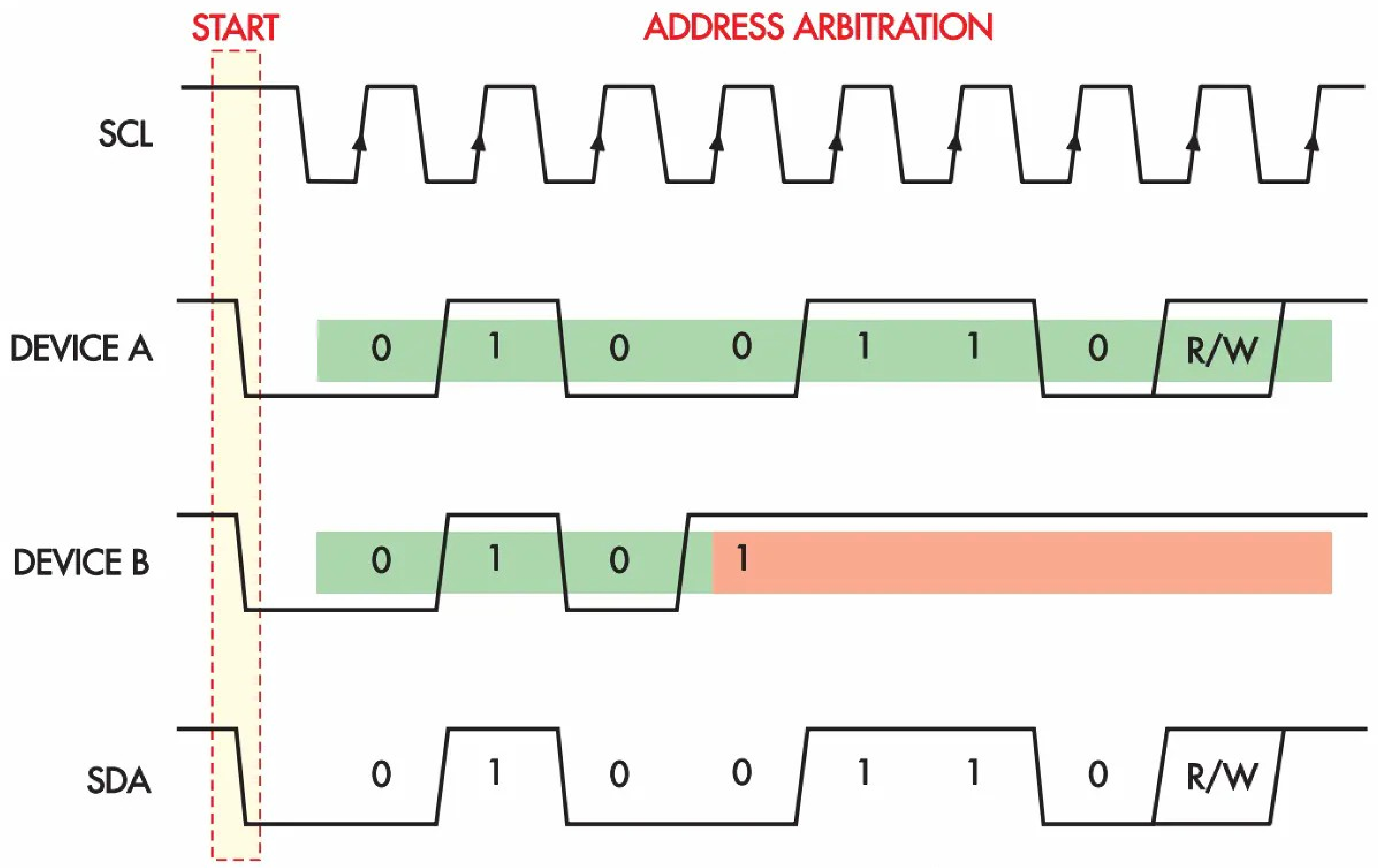


Fig.2: address arbitration occurs if two devices simultaneously attempt to write an address header to the bus. On the fourth clock cycle, device A's zero overrules device B's one; device B recognises this and gives up. Because zero-bits are asserted actively on the bus, while ones are passive, the device with the lowest address always wins arbitration.

Address arbitration

All communications on the I3C bus begin with the bus in the idle state, where the SDA and SCL lines are both high, and with open-drain drivers enabled.

The start of a transaction is indicated by a start condition, a high-to-low transition on SDA while SCL remains high. A transaction is terminated with a stop condition, a low-to-high transition on SDA with SCL high.

All other SDA transitions occur with SCL low. Repeated start conditions may be issued to allow multiple messages per transaction. A target address header follows each start or repeated start condition.

If all this sounds familiar, that's because, so far, this is identical to how I²C works. One key difference is that, with I3C, it is possible for targets to issue a start condition and/or to emit the address header under certain circumstances. On an I²C bus, only the controller can do that.

This can happen when a target wishes to signal an interrupt to the controller, a device wants to 'hot join' the bus, or a target wishes to become the controller. It is therefore possible that two or more devices may try to write the address header to the bus at the same time. We need a mechanism to arbitrate this conflict.

The open-drain nature of the bus during this phase provides a form of natural arbitration, ensuring that the device with the lowest address always 'wins'. Consider the example shown in Fig.2. Here, devices A and B, with hexadecimal addresses 0x26 (binary 010 0110) and 0x29 (binary 010 1001), respectively, attempt to write their address headers (7-bit address + read/write bit) to the bus.

On the first clock cycle after the start condition, each emits a logic zero by pulling the bus down with their open-drain driver. Both A and B monitor the bus and check that its level matches the value they emitted. If so, they each continue to deliver the header.

Both devices emit a logic one on the second clock cycle, and the passive pull-up pulls the bus high. Again, each device sees that the bus state is as expected and moves on to the next bit.

This process continues until the 4th bit, when device A emits a zero and device B emits a one. The bus will be pulled low by device A's open-drain driver, so device B will see a mismatch between the logic level it emitted and that which appeared on the bus. At this point, device B has 'lost' the arbitration and ceases to participate.

Meanwhile, device A continues to place its address on the bus unopposed, and the transaction it initiated ensues. Under these circumstances, device B will likely reattempt to assert the header after the transaction concludes and the bus is idle once again.

This type of arbitration is zero-dominant, so the device with the lowest address always wins. For this reason, most controller-initiated I3C transactions don't begin with a target address header as they do in I²C. If they did, it would not be possible for the target with the highest address to win an arbitration.

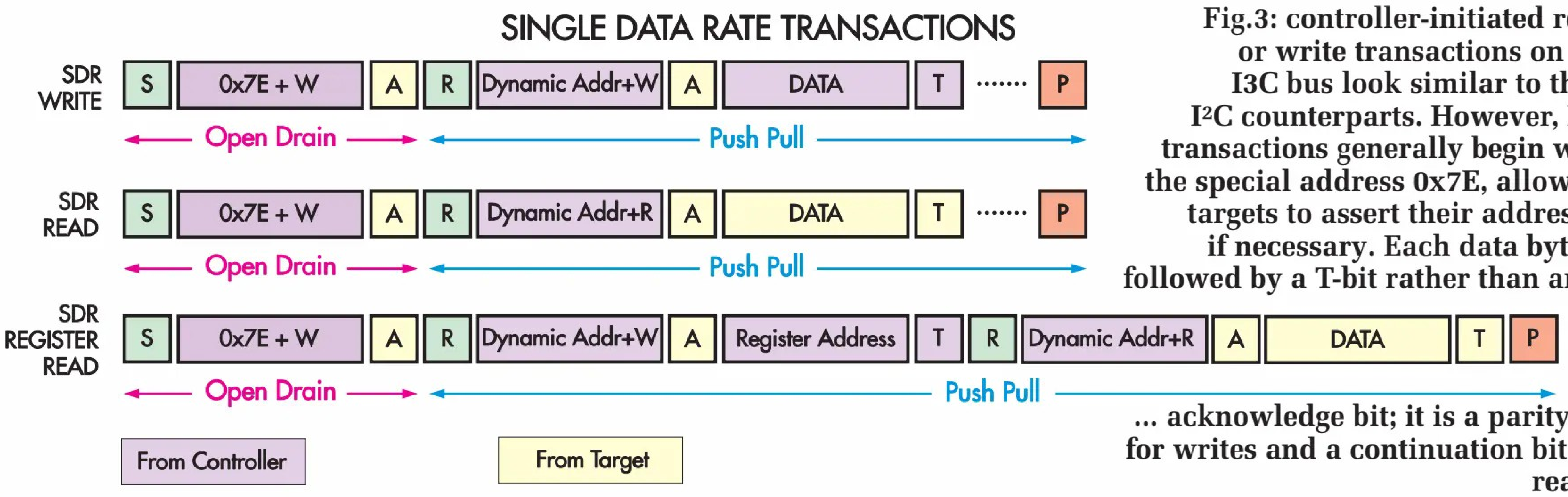


Fig.3: controller-initiated read or write transactions on the I3C bus look similar to their I²C counterparts. However, I3C transactions generally begin with the special address 0x7E, allowing targets to assert their addresses if necessary. Each data byte is followed by a T-bit rather than an ... acknowledge bit; it is a parity bit for writes and a continuation bit for reads.

Instead, they generally begin with the special address 7E hexadecimal (111 1110 binary). This address is higher than any valid target address, so it is guaranteed to lose arbitration to any target, should there be a conflict.

Single data rate transactions

I3C supports several different transaction types. Let's first look at a simple example where an I3C controller wishes to write to or read from a target, as illustrated at the top and centre of Fig.3, respectively.

SDR transactions begin from the idle state, with the controller issuing a start condition followed by an address header containing the special 7-bit address 0x7E and a write (zero) bit. All I3C targets will acknowledge (ACK) this address by pulling SDA low on the 9th clock cycle.

The controller then emits a repeated start condition followed by a non-arbitrable address header containing the target's dynamic address, along with the read or write bit, depending on the desired data direction. The target with a matching dynamic address will acknowledge (ACK), and any others will ignore the rest of the transaction.

Apart from this ACK, the whole transaction following the repeated start is carried out in push-pull mode with a nominal 12.5MHz clock. The controller or the target then begins sending data bytes, depending on the value of the read/write bit.

Because the sender is in push-pull mode, it is not possible for the receiving device to ACK each byte, as happens in I²C. Instead, the sending device sends a 'T-bit' on the 9th clock cycle.

For writes, the T-bit sent by the controller is a parity bit protecting the preceding byte. Odd parity is used; The parity bit is set or cleared such that the total number of ones in the data and the parity bit is odd.

In the case of reads, the target sets the T-bit to indicate that more bytes of data will come. The last byte has its T-bit cleared.

Fig.3 also shows a typical hybrid write-read that might be used to address and then read a specific register in a target. The arbitrable 0x7E header is written as before, followed by a repeated start condition and a write of the register address to the target. Another repeated start condition is then issued, followed by a read to obtain the register value; in this case, a single byte.

Apart from the T-bit replacing the acknowledge bit, and the clock speed, this is all very similar to I²C.

In-band interrupts

One elegant feature of I3C is the in-band interrupt (IBI) capability. A target may signal an interrupt by placing its dynamic address (and the read/write bit set to one) on the bus following a start condition. It may do this when the controller or another target initiates a start condition, or it may emit the start condition itself.

Since a controller message generally begins with the 0x7E address, the interrupting target will win the arbitration. If two or more targets interrupt simultaneously, the lowest addressed one will win. Fig.4 shows the typical IBI process.

The controller may accept the interrupt by ACKing the request as shown at the top or decline it by NACKing it as shown below that.

If the controller accepts the interrupt, it reads in a "mandatory byte" (MDB) sent by the interrupting target with details of the interrupt source. The target may optionally send additional bytes of data, indicated by the value of the T-bit. Once finished, the controller terminates the transaction with a stop condition.

If the controller denies the IBI request, the target may continue to try interrupting until the controller accepts. The controller may prevent a target from interrupting by sending a specific command to the target.

Common command codes

This introduces another feature of the I3C standard – the ability to send predefined common command codes (CCC) to targets. All targets must support a subset of CCCs, but others are

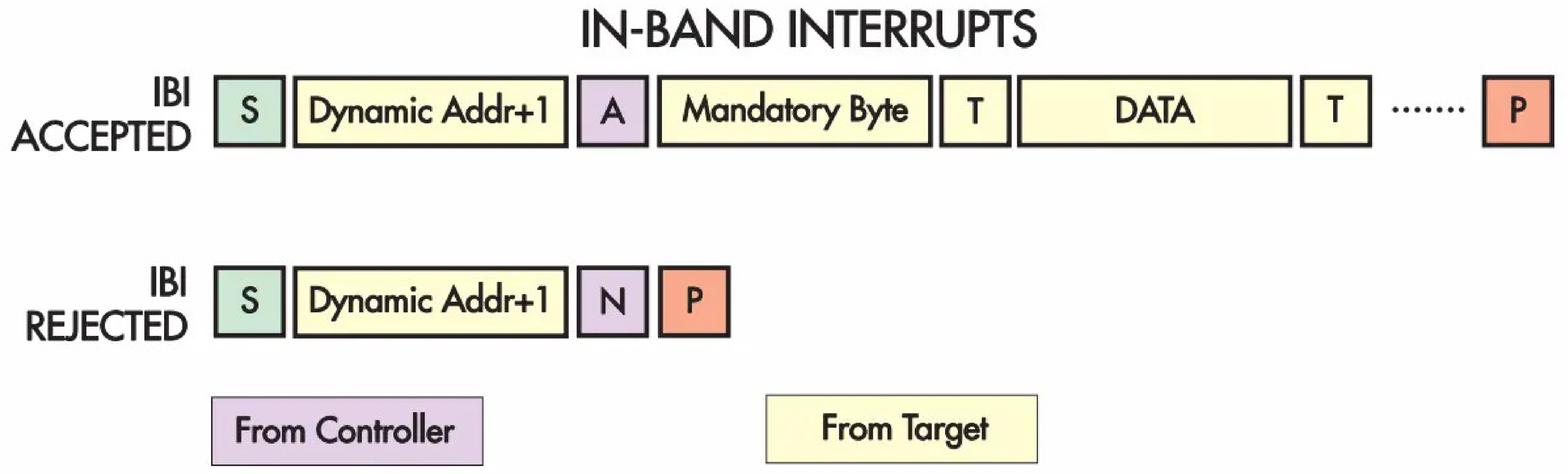


Fig.4: a target may signal an In-band Interrupt (IBI) by asserting its address header after a start condition. The controller accepts or rejects the IBI request by ACKing or NACKing the address. If accepted, the controller reads a Mandatory Byte describing the interrupt reason; the target may send further data after that.

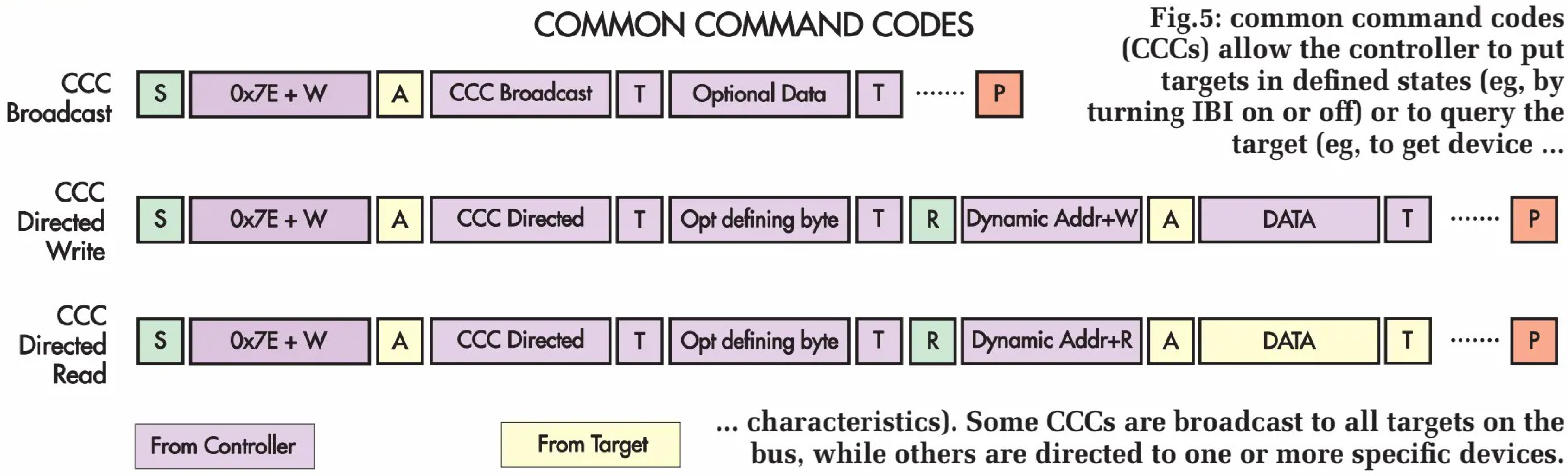


Fig.5: common command codes (CCCs) allow the controller to put targets in defined states (eg, by turning IBI on or off) or to query the target (eg, to get device ... ... characteristics). Some CCCs are broadcast to all targets on the bus, while others are directed to one or more specific devices.

optional or only required when the target has specific features.

All CCCs are one byte long, with those in the range 0x00 to 0x7F being broadcast codes that all devices will respond to, and those in the range 0x80 to 0xFF being direct codes intended for only a specific target. The standard defines many CCCs, a selection of which are included in Table 1.

Fig.5 shows how broadcast or direct CCCs are used. Broadcast CCC transmissions begin with the special address 0x7E, followed by the command code and any associated data bytes.

Direct CCCs begin the same way, with a 0x7E address followed by the CCC. This time, there may be a 'defining byte' associated with the CCC. A repeated start condition is then issued, followed by the dynamic address of the target to which the CCC is directed, plus a read/write bit. Additional targets may be addressed by issuing another repeated start condition and target address header.

Dynamic addressing

So far, we have referred to target addresses as though they are fixed, like I^{2}C device addresses. In I3C, the controller must assign each target a 7-bit dynamic address before any directed transactions can occur. The target retains this address until it is reset or the target receives a Reset Dynamic Address Assignment (RSTDAA) CCC.

Dynamic addressing has a couple of advantages over fixed addressing. Firstly, since lower addresses have higher arbitration priority, the controller can assign interrupt priorities by appropriate dynamic address choices. Secondly, it avoids conflicts between devices with identical static addresses, as occurs with I^{2}C.

That it can happen is unsurprising since there are only about 120 non-reserved 7-bit addresses and many thousands of unique devices. For example, the site https://i2cdevices.org/addresses suggests the address 0x44 is shared by no fewer than 19 devices, ranging from an IR temperature sensor to a high-side current monitor to a resistive touchscreen controller. Dynamic addressing avoids such overlaps.

Table 1 – some of the more frequently used I3C Common Control Codes

CCC	Type		Value	Brief Description
ENEC	Enable Events Command	Broadcast Write Direct Write	0x00 0x80	Enable Target events such as Hot-Join and In-Band Interrupt
RSTDAA	Reset Dynamic Address Assignment	Broadcast Write Direct Write	0x06 0x86	Discard current Dynamic Address and wait for new assignment
ENTDAA	Enter Dynamic Address Assignment	Broadcast Write	0x07	Enter Controller initiation of Dynamic Address Assignment Procedure
SETDASA	Set Dynamic Address from Static Address	Direct Write	0x87	Controller assigns a Dynamic Address to a Target with a known Static Address
SETNEWDA	Set New Dynamic Address	Direct Write	0x88	Controller assigns new Dynamic Address to a Target
SETMWL	Set Maximum Write Length	Broadcast Write Direct Write	0x09 0x89	Controller sets maximum write length
SETMRL	Set Maximum Read Length	Broadcast Write Direct Write	0x0A 0x8A	Controller sets maximum read length and IBI payload size
GETMWL	Get Maximum Write Length	Direct Read	0x8B	Controller queries Target's maximum possible write length
SETBUSCON	Set Bus Context	Broadcast Write	0x0C	Controller specifies a higher-level protocol and/or I3C specification version
GETMRL	Get Maximum Read Length	Direct Read	0x8C	Controller queries Target's maximum possible read length and IBI payload size
GETPID	Get Provisional ID	Direct Read	0x8D	Controller queries Target's Provisional ID
GETBCR	Get Bus Characteristics Register	Direct Read	0x8E	Controller queries Target's Bus Characteristics Register
GETDCR	Get Device Characteristics Register	Direct Read	0x8F	Controller queries Target's Device Characteristics Register
ENTHDR0	Enter HDR Mode 0	Broadcast Write	0x20	Controller has entered HDR-DDR Mode
ENTHDR1	Enter HDR Mode 1	Broadcast Write	0x21	Controller has entered HDR-TSP Mode
ENTHDR2	Enter HDR Mode 2	Broadcast Write	0x22	Controller has entered HDR-TSL Mode
ENTHDR3	Enter HDR Mode 3	Broadcast Write	0x23	Controller has entered HDR-BT Mode
RSTACT	Target Reset Action	Broadcast Write Direct Write & Read	0x2A 0x9A	Controller configures and/or queries Target Reset action and timing
GETSTATUS	Get Device Status	Direct Read	0x90	Controller queries Target's operating status
GETMXDS	Get Maximum Data Speed	Direct Read	0x94	Controller queries Target's maximum read/write data speeds and maximum read turnaround time

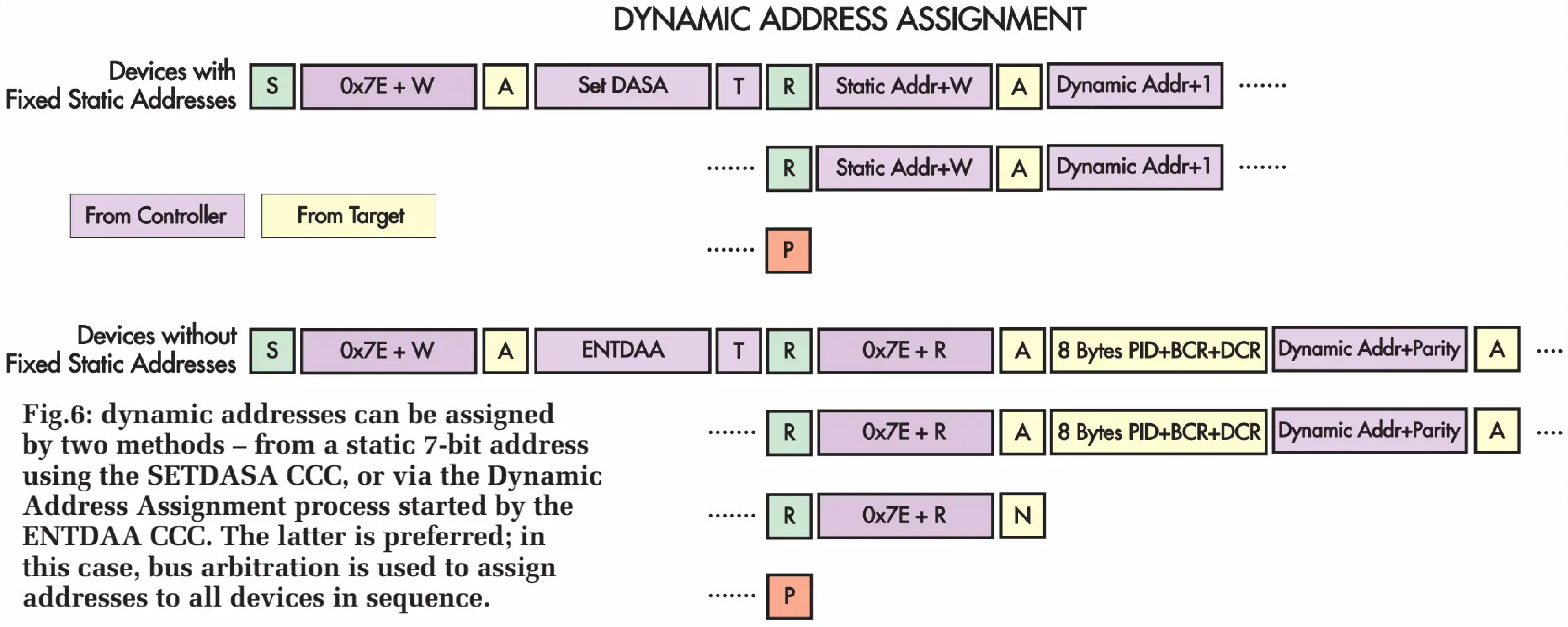

DYNAMIC ADDRESS ASSIGNMENT
Devices with Fixed Static Addresses
S
0x7E + W
A
Set DASA
T
R
Static Addr+W
A
Dynamic Addr+1
R
Static Addr+W
A
Dynamic Addr+1
From Controller
From Target
P
Devices without Fixed Static Addresses
S
0x7E + W
A
ENTDAA
T
R
0x7E + R
A
8 Bytes PID+BCR+DCR
Dynamic Addr+Parity
A
R
0x7E + R
A
8 Bytes PID+BCR+DCR
Dynamic Addr+Parity
A
R
0x7E + R
N
P

Fig.6: dynamic addresses can be assigned by two methods – from a static 7-bit address using the SETDASA CCC, or via the Dynamic Address Assignment process started by the ENTDAA CCC. The latter is preferred; in this case, bus arbitration is used to assign addresses to all devices in sequence.

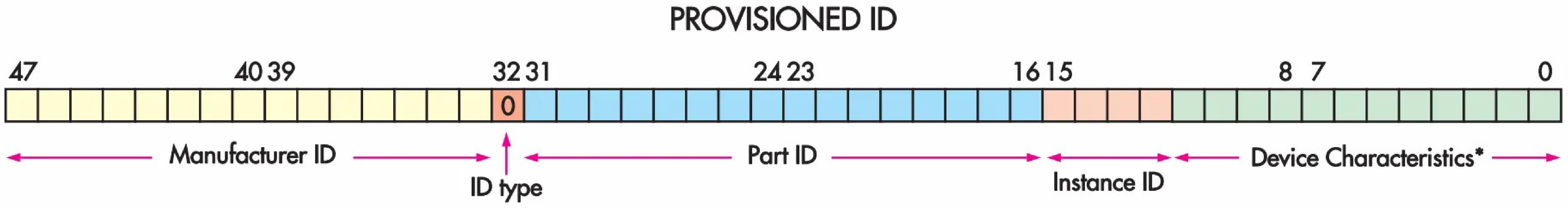

PROVISIONED ID
47
40 39
32 31
24 23
16 15
8 7
0
0
Manufacturer ID
ID type
Part ID
Instance ID
Device Characteristics*

Fig.7: the Provisioned Identifier (PID) is a 48-bit value containing a manufacturer ID, a part ID, an Instance ID and a 12-bit device characteristic descriptor. On some parts, the Instance ID is programmable via pins or other means to allow multiple instances of the same device to be uniquely identified on the bus.

There are two different options for assigning dynamic addresses. Some early devices do have a static 7-bit address, which is used to assign a dynamic address using the CCC "Set Dynamic Address from Static Address" (SETDASA), as shown at the top of Fig.6. The static address can't be used for any other purpose.

The controller starts the transaction in the usual way, sending the SETDASA CCC, followed by a restart, then the static address of the target with the read/write bit set. When the target ACKs the static address, the controller sends the dynamic address it wishes to assign.

The controller can optionally set further dynamic addresses by repeating the process after issuing a repeated start condition.

This approach does not really help with the problem of address duplication, so the preferred option is to assign dynamic addresses based on a 48-bit device identifier known as a 'Provisioned ID' (PID), the format of which is shown in Fig.7.

The upper 15 bits of this are the manufacturer ID assigned by the MIPI Alliance. The next bit, #32, is usually zero, while the following 16 bits are the part identifier. Bits 12 through 15 are an instance ID. These may be programmed by setting device pin(s) to specific levels, to allow the user to deploy multiple instances of the same device on the same bus.

The process for assigning dynamic addresses to these targets relies on the same bus arbitration mechanism we saw earlier. The lower part of Fig.6 shows this in action. The CCC Enter Dynamic Addressing Assignment (ENTDASA) is sent, followed by a repeated start condition and another 0x7E address.

Any targets on the bus that do not have a dynamic address assigned will ACK this and begin writing their 48-bit PID, followed by the contents of two 8-bit capability registers, BCA and DCA. This write is arbitrable, so only the device with the lowest PID will complete the process, at which time the controller sends the dynamic address it wishes to assign.

If the device acknowledges, the dynamic address is considered assigned. A repeated start condition is issued, and the whole process is repeated for the remaining targets with unassigned dynamic addresses until none are left. At that point, the 0x7E address is passively NACKed, and the assignment process is terminated.

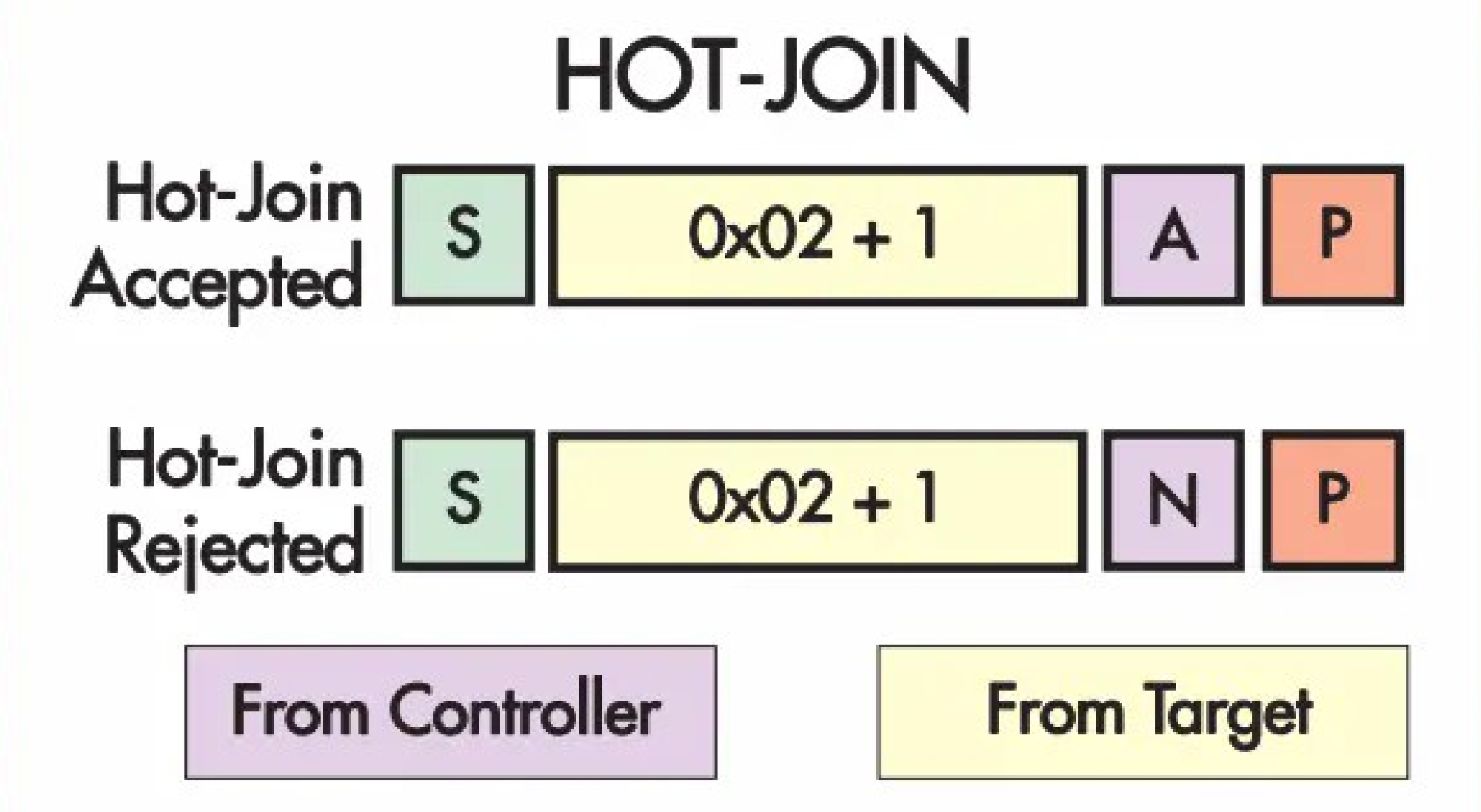

HOT-JOIN
Hot-Join Accepted
S
0x02 + 1
A
P
Hot-Join Rejected
S
0x02 + 1
N
P
From Controller
From Target

Fig.8: the hot join process is similar to the IBI process, except that the joining device uses the special 0x02 address. If the controller ACKs the request, the target waits for a dynamic address to be assigned by the controller.

Hot-Join mechanism

A target may join an I3C bus once it is up and running (for example, by being plugged in or powered up). It does this using a mechanism similar to the In-Band Interrupt, shown in Fig.8. The hot-joining device emits an address header with the special high-priority address 0x02, guaranteeing it will be heard.

The controller may accept the hot-join request by ACKing it or reject it by NACKing it. If the request is accepted, the joining device waits for the controller to assign a dynamic address via

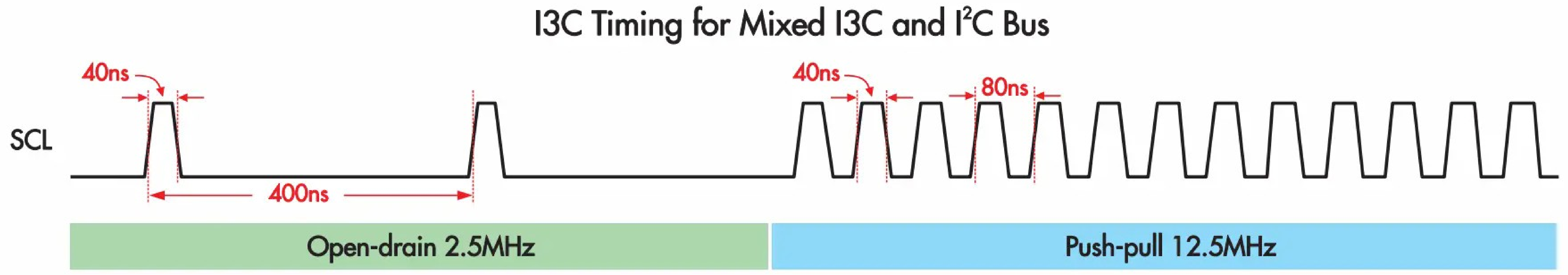


Fig.9: the I3C controller produces the SCL clock with a duty cycle such that the high period is 40ns or less. I^{2}C devices are required to have a 50ns glitch filter on their SCL and SDA inputs, so the I3C clock should be invisible to them.

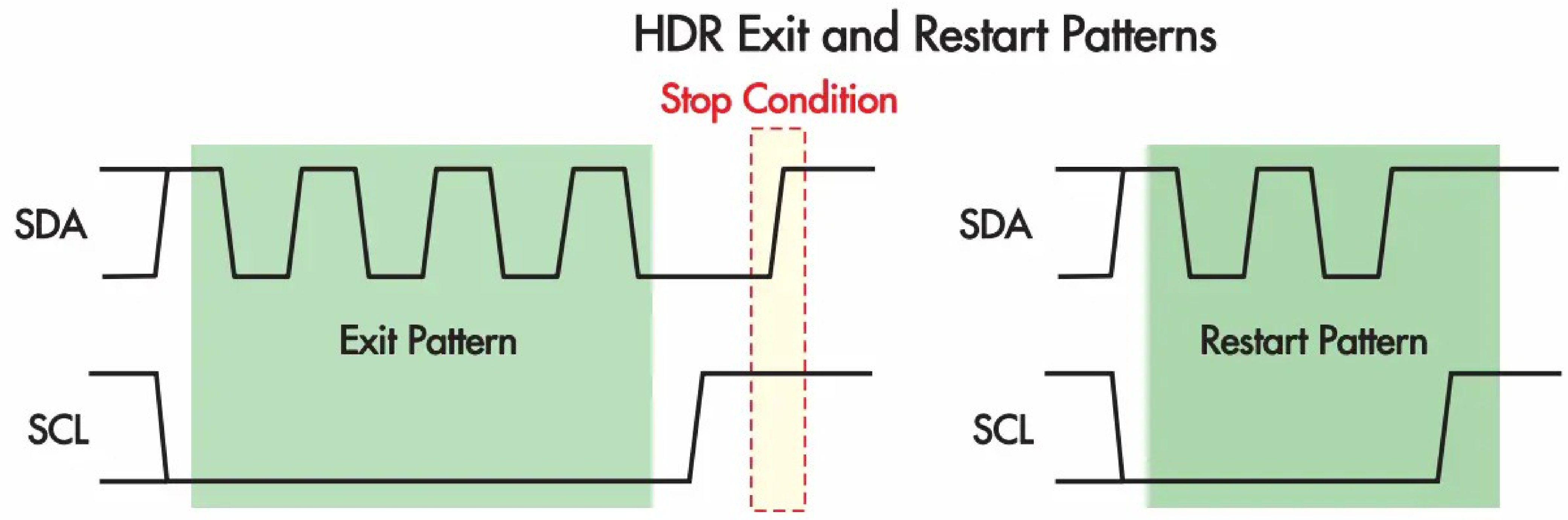


Fig.10: the HDR exit and restart patterns; the exit pattern is always followed by a stop condition and returns the bus to the idle state. The restart pattern is analogous to the repeated start condition and delineates multiple messages within an HDR transaction.

the ENTDAA CCC process described above. If the controller rejects the request, the joining device will likely continue to make requests unless the controller switches off hot-join requests via the DISEC CCC.

Note that this process works if more than one device joins simultaneously – the 0x02 address will be emitted by all devices simultaneously, and they will all succeed in the arbitration. The controller ACK will put them all in a state awaiting dynamic address assignment, and the ENTDAA assignment process will assign them all addresses in order of their PIDs.

I^{2}C compatibility

I mentioned before that I^{2}C targets can co-exist with I3C targets on the same bus. It is important that these devices don't react to I3C messages and potentially disrupt them. I3C uses a couple of mechanisms to reduce the chance of that happening. Firstly, the special address 0xFE in the header of I3C messages is a reserved address in I^{2}C and should be ignored by all complying devices.

On top of this, I3C takes advantage of the 50ns glitch filter that is required by the standard on the SDA and SCL pins of I^{2}C devices. The duty cycle of the SCL signal is managed so that the clock high period is always less than or equal to 40ns, making the I3C clock 'invisible' to I^{2}C targets. They should ignore any signal on their SDA line, because they don't see any clock transitions on SCL.

Fig.9 shows how this works. For the 12.5MHz push-pull clock, the high and low pulses are naturally 40ns long at a 50% duty cycle. The 2.5MHz open-drain clock is emitted at a duty cycle of around 10%, leaving the high pulses 40ns long.

This behaviour is only needed if there are legacy devices on the bus. Most controllers allow for a 50% duty cycle to be used on both clocks for 'pure' I3C buses.

When communicating with I^{2}C devices, the SCL clock is reduced to 400kHz or 1MHz with a 50% duty cycle and transactions are carried out in open-drain mode with signalling conforming to the I^{2}C standard. However, I3C does not support the I^{2}C features of 10-bit addressing, clock stretching or multi-master operation, so be careful.

High data rate modes

So far, everything we have covered is occurring in 'Single Data Rate' (SDR) mode. The I3C protocol supports several higher data rate (HDR) modes. I am only going to cover the most common one of those in the interests of brevity. Regardless of the HDR mode chosen, the I3C bus is always initialised and configured in SDR mode.

Only limited extra functionality is available in the HDR modes – you can't assign dynamic addresses or receive in-band interrupts, for example.

HDR modes are always entered from SDR mode by issuing the appropriate Enter HDR Mode X (ENTHDRx) CCC. Once entered, the HDR mode has a bus-wide effect until it is exited via an HDR exit pattern. Within an HDR mode transaction, multiple messages can be separated by HDR restart

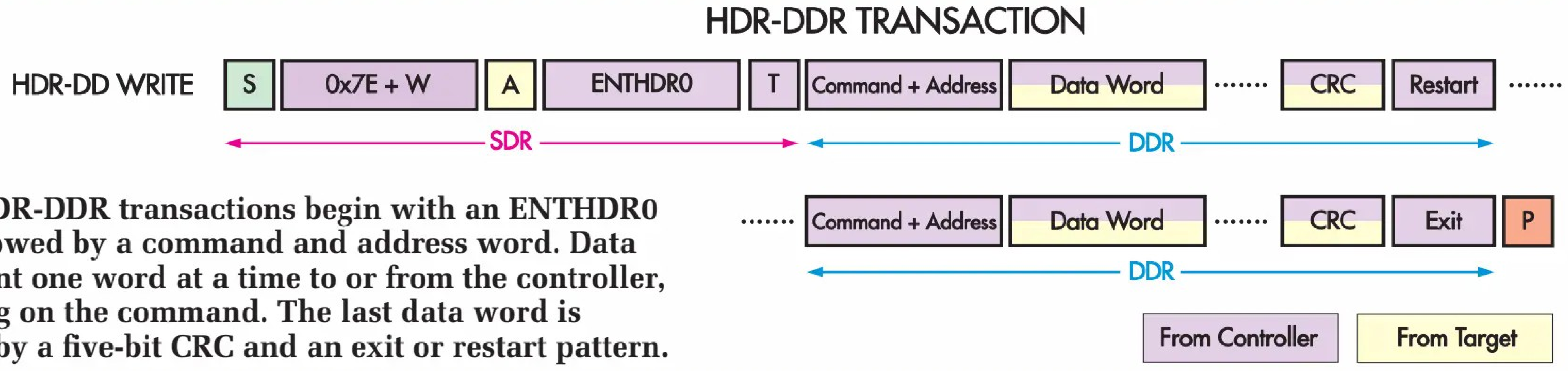


Fig.11: HDR-DDR transactions begin with an ENTHDR0 CCC, followed by a command and address word. Data is then sent one word at a time to or from the controller, depending on the command. The last data word is followed by a five-bit CRC and an exit or restart pattern.

patterns, analogous to the repeated start condition in SDR mode.

The HDR exit and restart patterns are shown in Fig.10. The exit pattern is defined as four consecutive falling edges on SDA while SCL is low. It is always followed by a stop condition. The restart pattern is defined as two successive toggles of SDA (fall, rise, fall, rise) followed by a rising edge on SCL.

HDR-DDR

The common HDR Double Data Rate (HDR-DDR) mode uses the same 12.5MHz clock as SDR mode but allows data bits to be sent on both rising and falling clock edges, effectively doubling the data throughput.

Fig.11 shows how a typical HDR-DDR transaction works. After sending the ENTHDR0 CCC in SDR mode, the bus switches to HDR-DDR mode. The controller then issues a command and address word indicating the data direction (read or write) and the intended target's dynamic address, followed by one or more data words written by either the controller or the target, depending on the data direction.

When all the data has been sent, the sender concludes with a five-bit CRC. The controller then emits a restart pattern if another DDR message is to be sent, or an exit pattern and stop condition if not.

The basic unit of transmission (except for the CRC) is a 16-bit word with a two-bit preamble and two trailing parity bits, for a total of 20 bits. Fig.12 shows how the command, data and CRC words are constructed.

The Command and CRC words begin with the same preamble but are distinguishable by context – the command word only ever comes immediately after entry to the HDR mode or a restart pattern, while the CRC only ever comes after a data word. Only a very limited range of CCCs are supported in HDR-DDR mode.

Other features

DDR mode can achieve an effective data throughput of 20Mbps, comparable with SPI (but with error checking). Faster HDR modes are possible, including some that allow for two or four data lanes and can achieve effective data rates of up to 96Mbps. However, not all modes are supported by all targets.

In addition to those features described here, I3C offers some other interesting capabilities, including the ability to reset targets (individually or in groups) over the bus and to send synchronising 'ticks' to targets to coordinate timing. It supports group addressing, where a set of targets share a group address (alongside their individual dynamic addresses), so they can be sent messages simultaneously.

It is also possible for one target to communicate directly with another via device-to-device tunnelling. There is a lot to this standard; it will be interesting to see what features receive support from chip vendors.

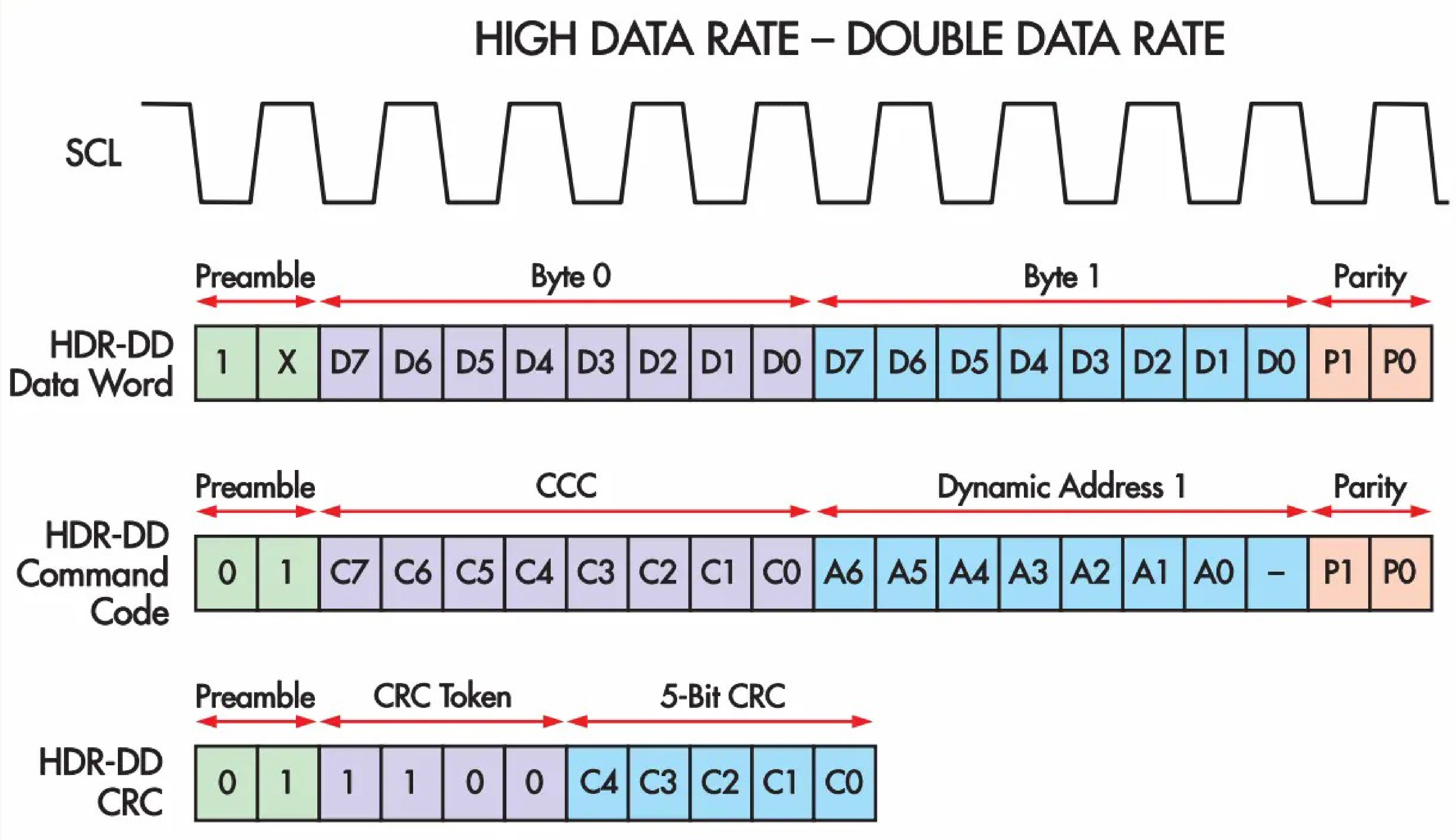


Fig.12: data is transmitted on both clock edges in HDR-DDR mode. For data and command words, a two-bit preamble is followed by 16 data bits and two parity bits. The CRC word is slightly different and is always followed by a restart or exit pattern.

Trying it out

I have experimented with I3C over the last couple of weeks, using an NXP LPC865 microcontroller with an integrated I3C controller. For targets, I tried a Bosch BMI323 inertial measurement unit (IMU), an ST Microelectronics LPS22HH humidity sensor and a TDK ICM42688P motion sensor.

I was able to test a lot of the SDR functionality, and it works as advertised, but the experience was not as smooth as it could have been. I think these rough edges are due to the relative immaturity of the standard. The device data sheets were a mixed bag regarding the thoroughness of their documentation of I3C features – I did have to resort to a bit of trial and error to get things going.

The MCU toolchain and the I3C driver provided in the SDK worked fine for my purposes. Debugging the bus was a challenge, as my logic analysers do not currently have support for the I3C protocol. For this reason, I had to resort to printing out waveforms and marking ones and zeros by hand on a few occasions.

I am sure that all of these points will improve as I3C becomes more common. Right now, I3C is available on only a limited number of devices, which are helpfully listed at https://binho.io/blogs/i3c-reference/i3c-devices

I believe this list will grow, given that the MIPI Alliance lists almost every silicon vendor as a member, and it has a pretty good track record of establishing standards. Over the next few years, I suspect we will see I3C becoming more and more popular.

References

I²C-Bus Specification and User Manual (2021): NXP, https://pemag.au/link/abtv

I3C and I3C Basic specifications: MIPI, https://pemag.au/link/abgm

ICM-42688-P: TDK, https://pemag.au/link/abtw

Bosch Sensortec BMI323 Inertial Measurement Unit (IMU): Bosch, https://pemag.au/link/abtx

Arm Cortex-M0+ LPC86x 60MHz 32-Bit Microcontrollers (MCUs): NXP, https://pemag.au/link/abty

STMicroelectronics LPS22HH High-Performance MEMS Nano Pressure Sensor datasheet: ST, https://pemag.au/link/abtz

PE

DOWNLOAD YOUR PE/EPE BACK ISSUES

COMPLETE-YEAR collections – all 12 issues for each year!

From just £5.95 each

VOL 28 – January to December 1999 – **£5.95**
VOL 29 – January to December 2000 – **£6.95**
VOL 30 – January to December 2001 – **£7.95**
VOL 31 – January to December 2002 – **£7.95**
VOL 32 – January to December 2003 – **£7.95**
VOL 33 – January to December 2004 – **£7.95**
VOL 34 – January to December 2005 – **£8.95**
VOL 35 – January to December 2006 – **£8.95**
VOL 36 – January to December 2007 – **£8.95**
VOL 37 – January to December 2008 – **£8.95**
VOL 38 – January to December 2009 – **£10.95**
VOL 39 – January to December 2010 – **£11.95**
VOL 40 – January to December 2011 – **£12.95**
VOL 41 – January to December 2012 – **£14.95**
VOL 42 – January to December 2013 – **£15.95**
VOL 43 – January to December 2014 – **£16.95**
VOL 44 – January to December 2015 – **£17.95**
VOL 45 – January to December 2016 – **£19.95**
VOL 46 – January to December 2017 – **£22.95**
VOL 47 – January to December 2018 – **£27.95**
VOL 48 – January to December 2019 – **£27.95**
VOL 49 – January to December 2020 – **£29.95**
VOL 50 – January to December 2021 – **£32.95**
VOL 51 – January to December 2022 – **£34.95**
VOL 52 – January to December 2023 – **£34.95 – NEW!**
VOL 53 – January to December 2024 – **£34.95 – NEW!**

FIVE-YEAR collections – 60 issues!

From just £14.95 each

PE2832: Jan 99 to Dec 03 – £14.95
PE2933: Jan 00 to Dec 04 – £14.95
PE3034: Jan 01 to Dec 05 – £14.95
PE3135: Jan 02 to Dec 06 – £14.95
PE3236: Jan 03 to Dec 07 – £14.95
PE3337: Jan 04 to Dec 08 – £14.95
PE3438: Jan 05 to Dec 09 – £19.95
PE3539: Jan 06 to Dec 10 – £25.95
PE3640: Jan 07 to Dec 11 – £25.95
PE3741: Jan 08 to Dec 12 – £25.95
PE3842: Jan 09 to Dec 13 – £33.95
PE3943: Jan 10 to Dec 14 – £33.95
PE4044: Jan 11 to Dec 15 – £33.95
PE4145: Jan 12 to Dec 16 – £35.95
PE4246: Jan 13 to Dec 17 – £37.95
PE4347: Jan 14 to Dec 18 – £38.95
PE4448: Jan 15 to Dec 19 – £39.95
PE4549: Jan 16 to Dec 20 – £42.95
PE4650: Jan 17 to Dec 21 – £44.95
PE4751: Jan 18 to Dec 22 – £44.95 – NEW!
PE4852: Jan 19 to Dec 23 – £44.95 – NEW!

20 YEARS OF PE!

AMAZING BUNDLE OFFER
Every issue from 1999 – 2018

240 magazines for just

£69.95

PE2832: Jan 99 to Dec 03
PLUS!
PE3337: Jan 04 to Dec 08
PLUS!
PE3842: Jan 09 to Dec 13
PLUS!
PE3447: Jan 14 to Dec 18

Download your collections at: www.electronpublishing.com

Constructional Project

By John Clarke

8-Channel Learning IR Remote Receiver

This eight-channel relay board can have its outputs switched on and off using almost any remote control, including universal types. Each output can be set to toggle on or off, switched on for a fixed period, or on while the button is held down. The outputs can be controlled by an onboard reed relay or a transistor; the latter can switch external relays.

With so many appliances operated using infrared (IR) remote controls, you are bound to have at least one remote that is not used anymore. With our 8-Channel Learning IR Remote Receiver, it can be put back in service to provide control over eight separate relay outputs to control low-voltage DC or AC devices.

Many different kinds of remote control can operate the Receiver; you can even use it with multiple remotes. It learns the remote control code to switch each of its eight outputs. You could use a different remote control unit for each output if you wanted to. Most people would use a single remote control, though.

Remote controls transmit signals using specific IR protocols. These are usually transmitted using an infrared LED that is modulated on and off at between 36kHz and 40kHz. The modulated signal is switched on and off in a pattern with a start code, followed by address and command codes (visible in Scopes 1 to 4).

The address determines what appliance the code is to control, such as a TV, satellite decoder, DVD player, amplifier etc. The command code indicates what function is to operate. This can be power on or off, channel selection, volume up, volume down, mute etc.

Our Receiver can be used with remotes that produce signals in the NEC, Sony, RC5 and RC6 remote control protocols. More information about these is in a panel overleaf titled "Infrared Coding". Many remotes will use one of those protocols.

The controller has eight separate outputs, and each one can be switched using a separate code. Each channel can either be controlled by a reed relay (normally open contacts) or an open-collector transistor. Reed relays can be used for all channels, open collector

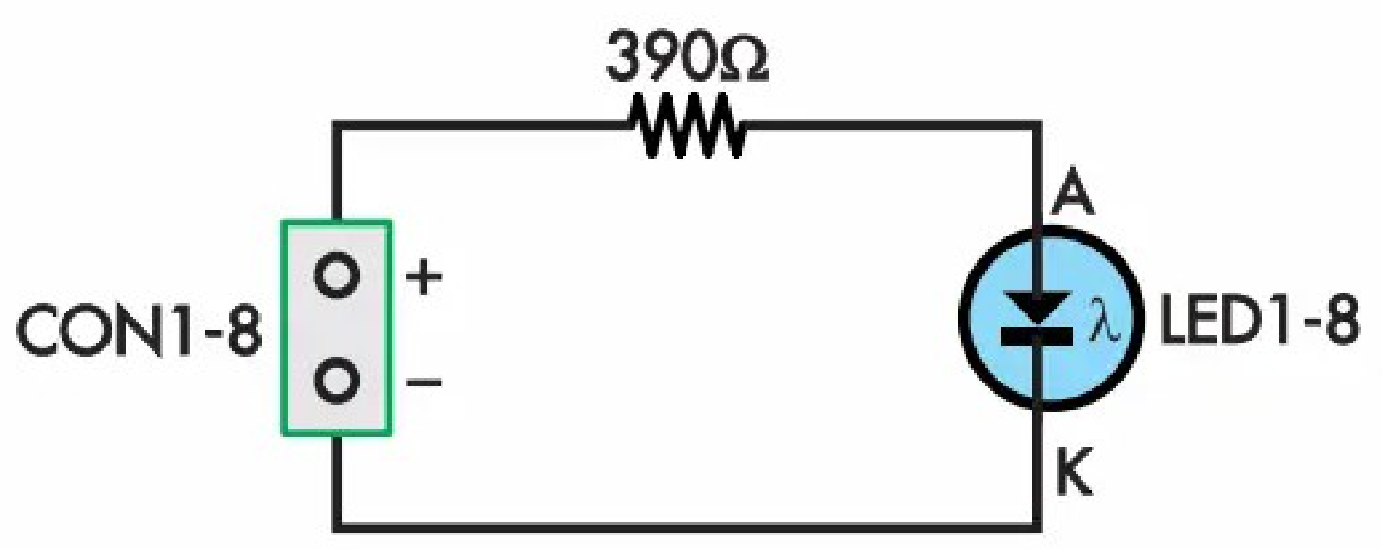


Fig.1: driving an external LED from an open-collector output. With a 12V supply, the 390Ω resistor will limit the current to around 25mA.

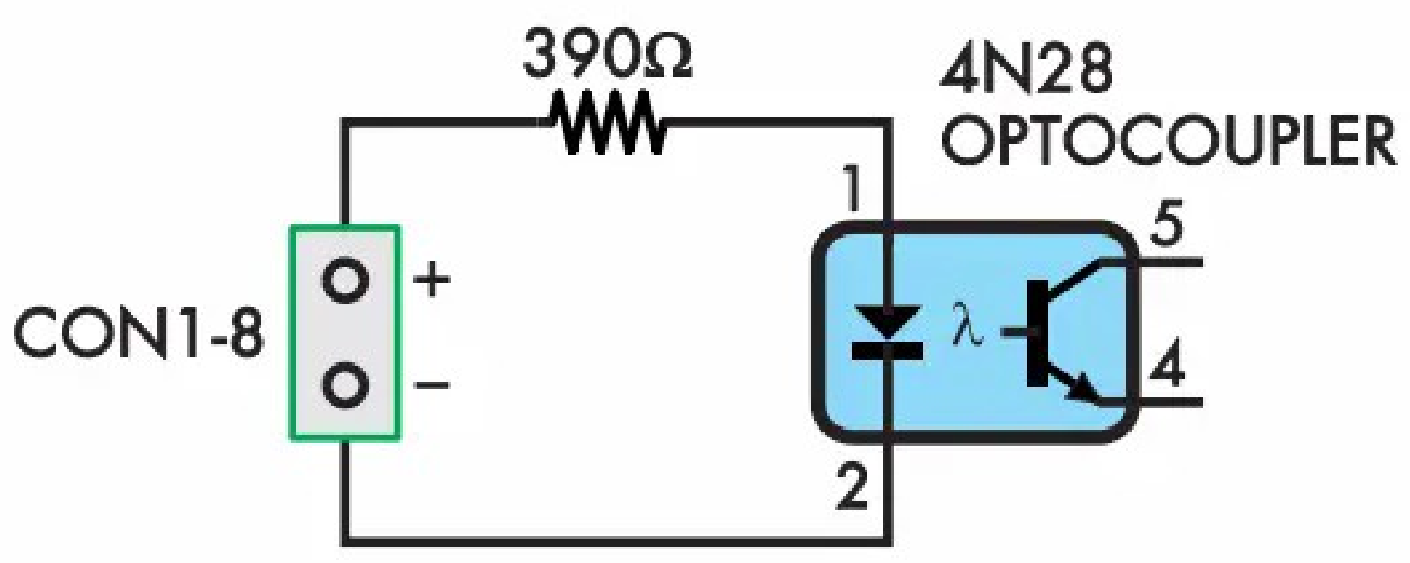


Fig.2: an opto-coupler's outputs are triggered by an internal LED, so driving them is basically the same as driving LEDs.

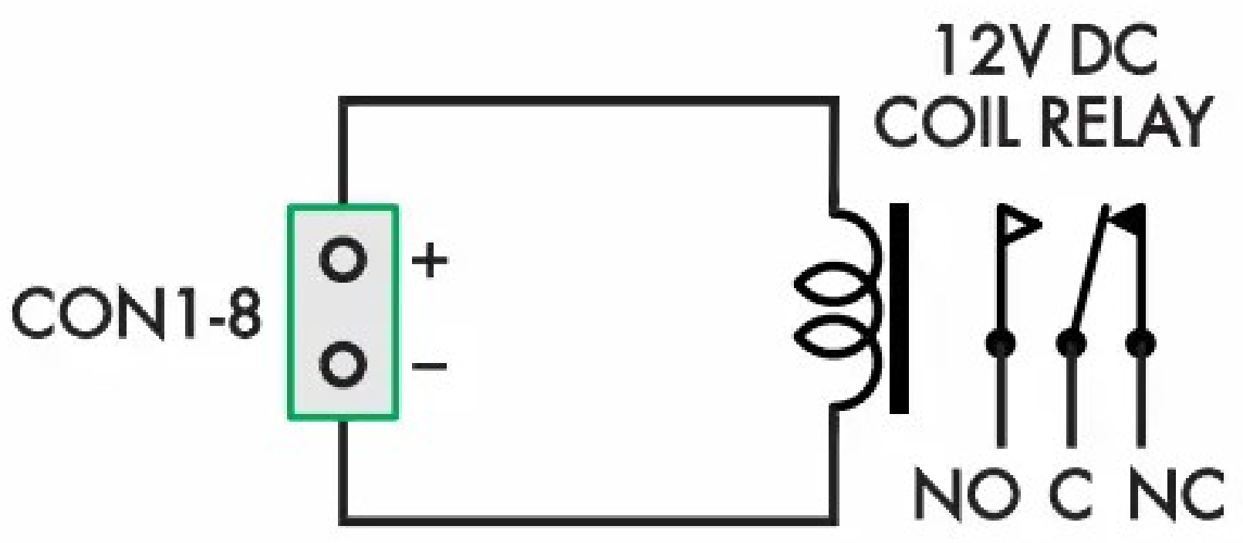


Fig.3: no series resistor is required if the coil is rated at 12V DC when driving an external relay from an open-collector output.

Features

Learns infrared remote control codes from a handheld IR remote

Supports four different IR protocols

1-8 output channels controlled by reed relays or open-collector transistors

Can be used with external relays (12V DC coil types)

Eight LED channel status indicators

Momentary or toggle operation on each output

Adjustable timer for momentary outputs (125ms to 32s)

Timer settings are shown on an 8-LED dot bargraph

Specifications

IR reception range: typically 10m

Power supply: 12V DC at 150mA+ (external relays may require more current)

Output switching: up to 24V @ 500mA

IR codes supported: learns NEC, Sony and Philips RC5/RC6 remote protocols

Momentary mode: 16 timer values, from 125ms to 32 seconds

Output toggle rate: minimum cycle time of 600ms

Oscillator frequency adjustment: ±6% in 128 steps

Power-on indication: dimmed LED

outputs for all channels or a mixture of the two.

Both output types can switch LEDs or other low-current loads. Alternatively, the transistor outputs can drive relays with 12V DC coils and contacts that can handle higher voltages and/or currents.

You don't need to build the controller with all eight outputs if you don't need them; just make it with fewer if that's all you need.

Outputs

The reed relays are ideal for switching low voltages (up to 24V maximum) and currents up to 500mA. They can be used to trigger pushbutton switches on equipment by wiring the reed relay contacts across the switch.

A reverse-biased diode should be connected across the relay's contacts if switching inductive loads. Never use the onboard reed relays to switch mains voltages directly. Neither the relays nor the PCB tracks can handle that. If you need to switch higher voltages, use the open-collector transistor outputs to switch appropriately-rated external relays.

Any external relays used for mains switching must be built to comply with mains voltage safety standards, including using correctly rated wire of the right colour and adequate insulation.

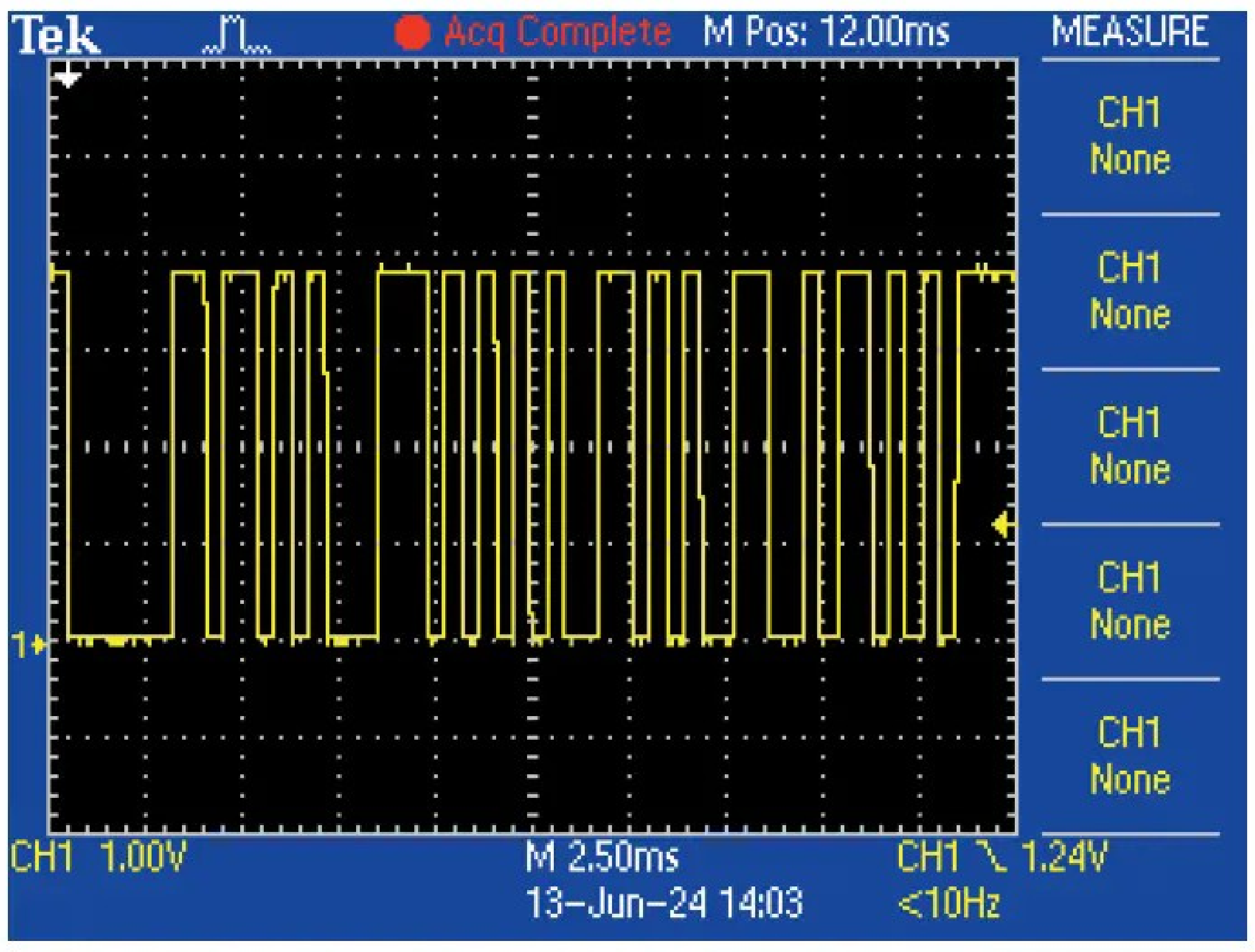
Figs.1-4 show a few different ways you can use the eight outputs when they are driven by open-collector transistors. Fig.1 shows how you can drive an external LED, Fig.2 shows how an external opto-coupler can be switched, Fig.3 shows how to drive an external relay and Fig.4 shows how you can switch off or control the direction of a motor.

With the motor, you can use the channels with the outputs set for momentary or toggle operation. In the momentary mode, pressing (and holding) the button for open-collector output X activates RELAY 1 and causes the motor to rotate one way, while pressing the button for output Y activates RELAY 2 and causes the motor to rotate the other way.

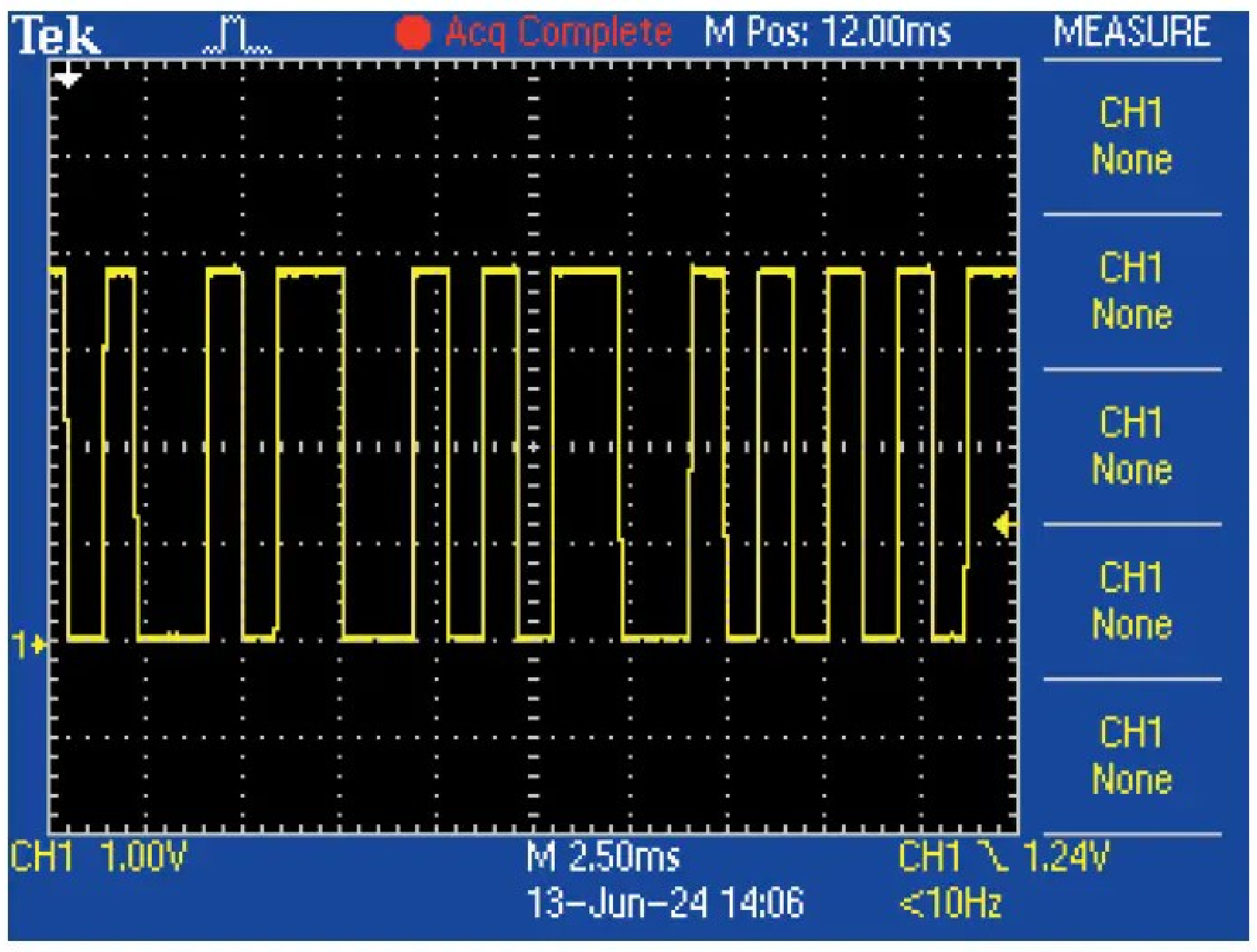
With both outputs set for toggle operation, the motor will be stopped until one of the outputs is toggled. Its direction of rotation will depend on which output is switched on. The motor can then be reversed by toggling both outputs, or stopped by toggling either output.

Remote control protocols

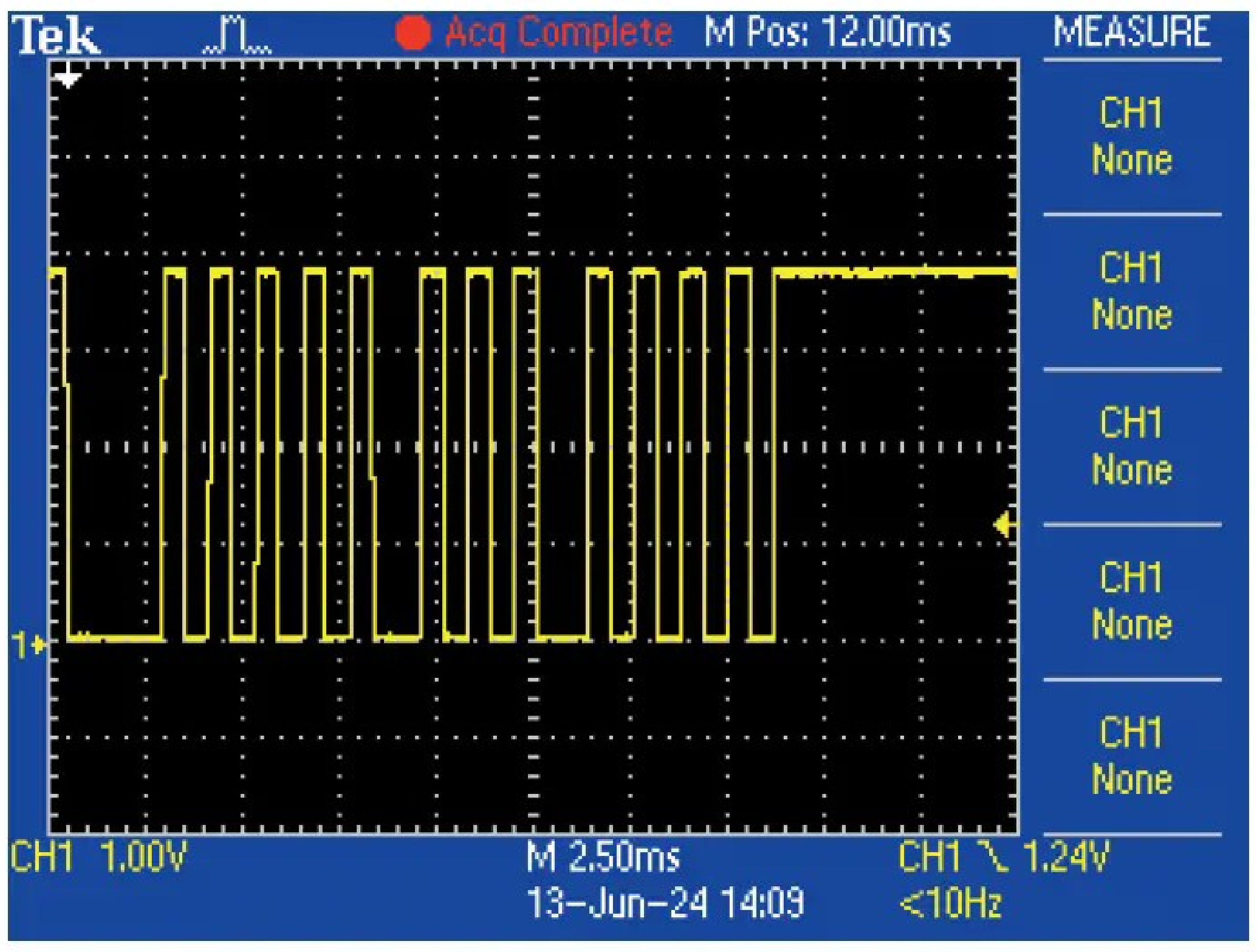
Scope grabs 1-4 show captured waveforms for decoded IR signals transmitted in the RC6, RC5, Sony and NEC


Tek
Acq Complete
M Pos: 12.00ms
MEASURE
CH1 None
CH1 None
CH1 None
CH1 None
CH1 None
CH1 1.00V
M 2.50ms
13-Jun-24 14:03
CH1 ↘ 1.24V
<10Hz

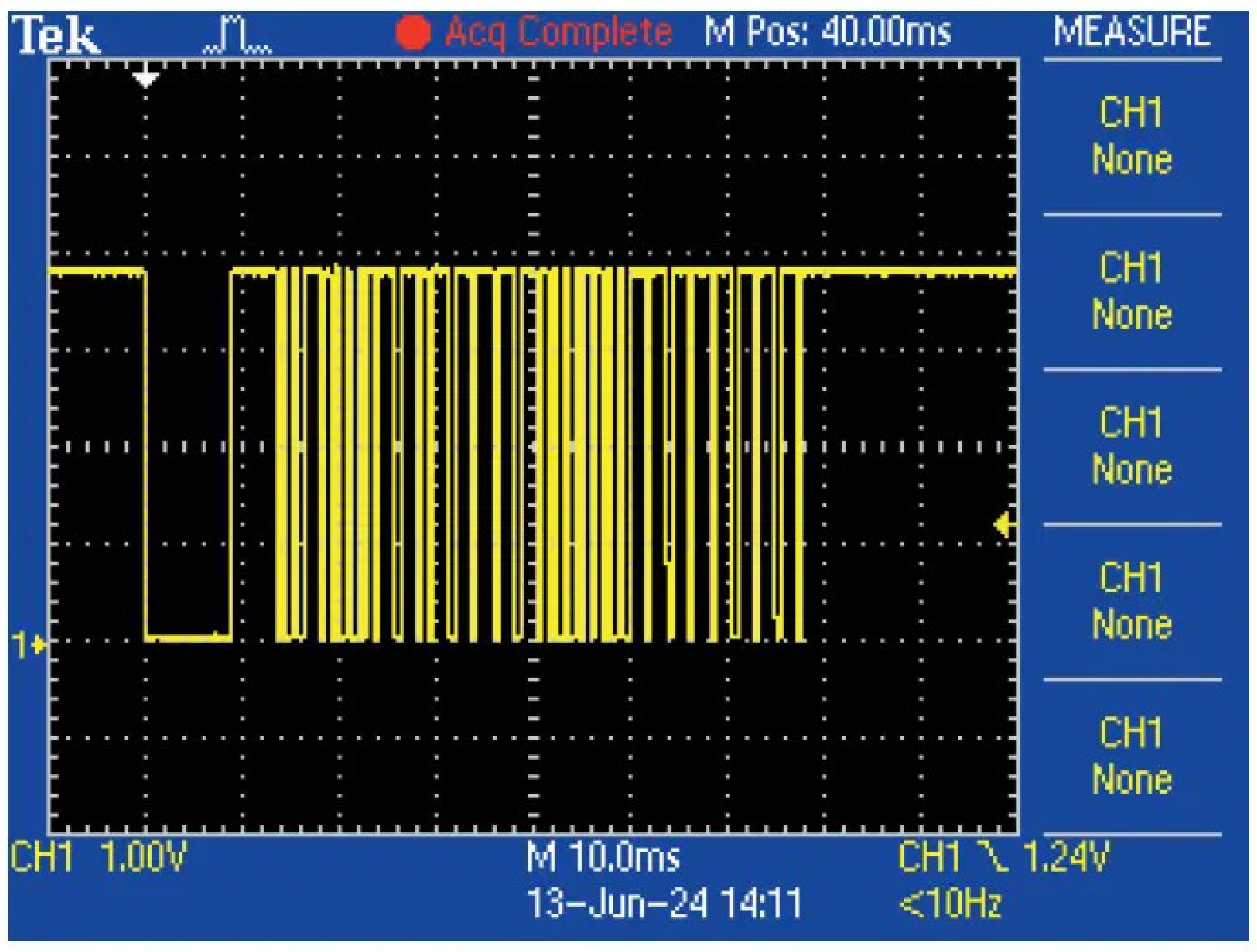
Scope 1: an oscilloscope capture of the output of IRD1 when receiving a Philips RC6-coded signal.


Tek
Acq Complete
M Pos: 12.00ms
MEASURE
CH1 None
CH1 None
CH1 None
CH1 None
CH1 None
CH1 1.00V
M 2.50ms
13-Jun-24 14:06
CH1 ↘ 1.24V
<10Hz

Scope 2: an oscilloscope capture of the output of IRD1 when receiving a Philips RC5-coded signal.


Tek
Acq Complete
M Pos: 12.00ms
MEASURE
CH1 None
CH1 None
CH1 None
CH1 None
CH1 None
CH1 1.00V
M 2.50ms
13-Jun-24 14:09
CH1 ↘ 1.24V
<10Hz

Scope 3: an oscilloscope capture of the output of IRD1 when receiving a Sony-coded signal.


Tek
Acq Complete
M Pos: 40.00ms
MEASURE
CH1 None
CH1 None
CH1 None
CH1 None
CH1 None
CH1 1.00V
M 10.0ms
13-Jun-24 14:11
CH1 ↘ 1.24V
<10Hz

Scope 4: an oscilloscope capture of the output of IRD1 when receiving an NEC-coded signal.

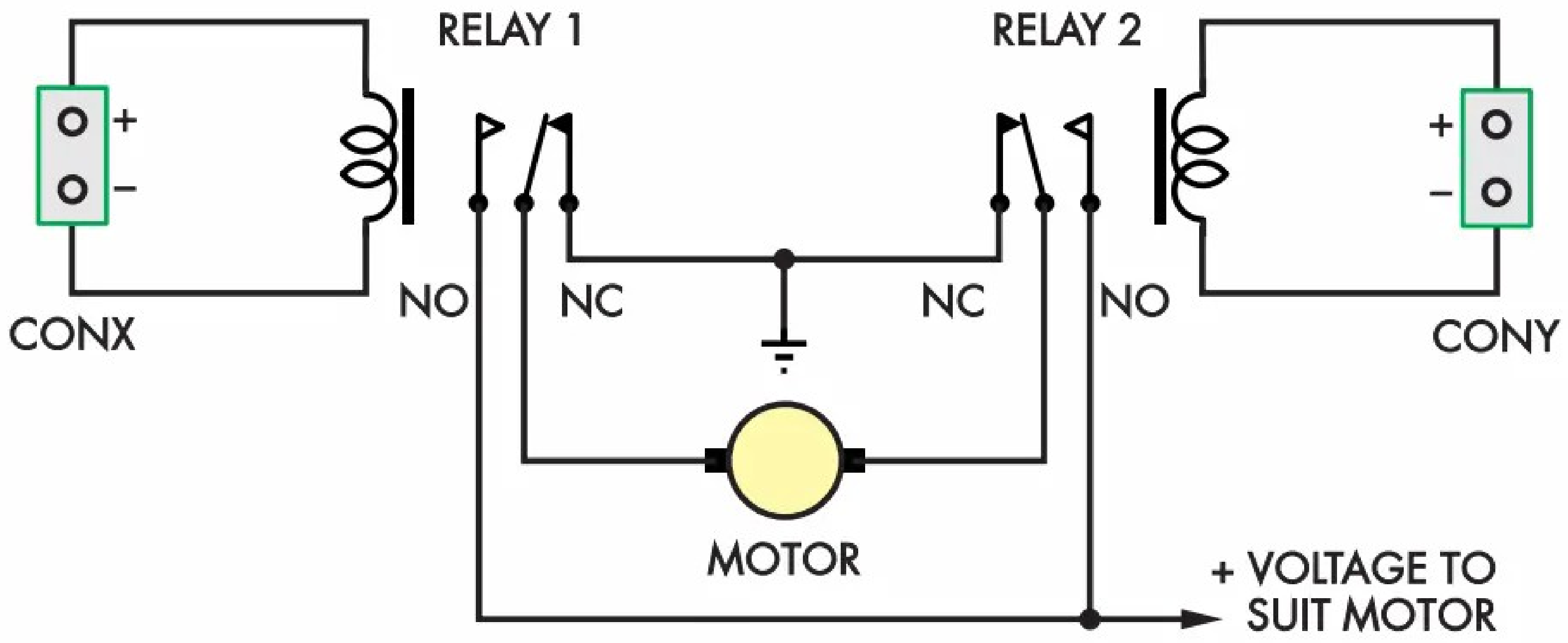

RELAY 1
RELAY 2
+
–
CONX
NO
NC
NC
NO
+
–
CONY
MOTOR
+ VOLTAGE TO SUIT MOTOR

Fig.4: a simple method to control the direction of a motor using two external relays, driven from two of the Receiver's outputs.

protocols, respectively. These waveforms were taken from the output of IRD1. The 36-40kHz modulation was removed by the receiver; its output is low during the modulated burst and high when there is a pause in modulation.

Scope 5 shows the repeat pulses for the NEC protocol that follow the initial main code if the remote control button is held down.

For the remaining protocols (RC5, RC6 and Sony), holding down the remote control button simply repeats the code that is initially sent. More details are provided in the “Infrared Coding” panel.

Momentary & toggle modes

Each output can be set for momentary or toggle operation. With the momentary selection, an output and its associated LED switch on when the remote control button is pressed, then off again after a set period from ⅛th of a second (125ms) to 32 seconds. The timer period can be elongated by holding down the remote control button, in which case the timer starts when the button is released.

In toggle mode, the output switches on with one press of an IR remote button, and it remains on until the same button is pressed again, whereupon it switches off.

During the IR code learning procedure, a pushbutton switch on the controller board selects momentary or toggle operation for each output. For channels set to momentary mode, the on-time period is set at the same time, using a trimpot, with the front panel LEDs indicating the period selected.

A panel on Infrared Coding

Most infrared controllers switch their LED on and off at a modulation frequency of 36-40kHz in bursts (pulses), with the length of and space between each (pauses) indicating which button was pressed. The series of bursts and pauses is in a specific format (or protocol) and there are several commonly used. This includes the Manchester-encoded RC5 and RC6 protocols originated by Philips.

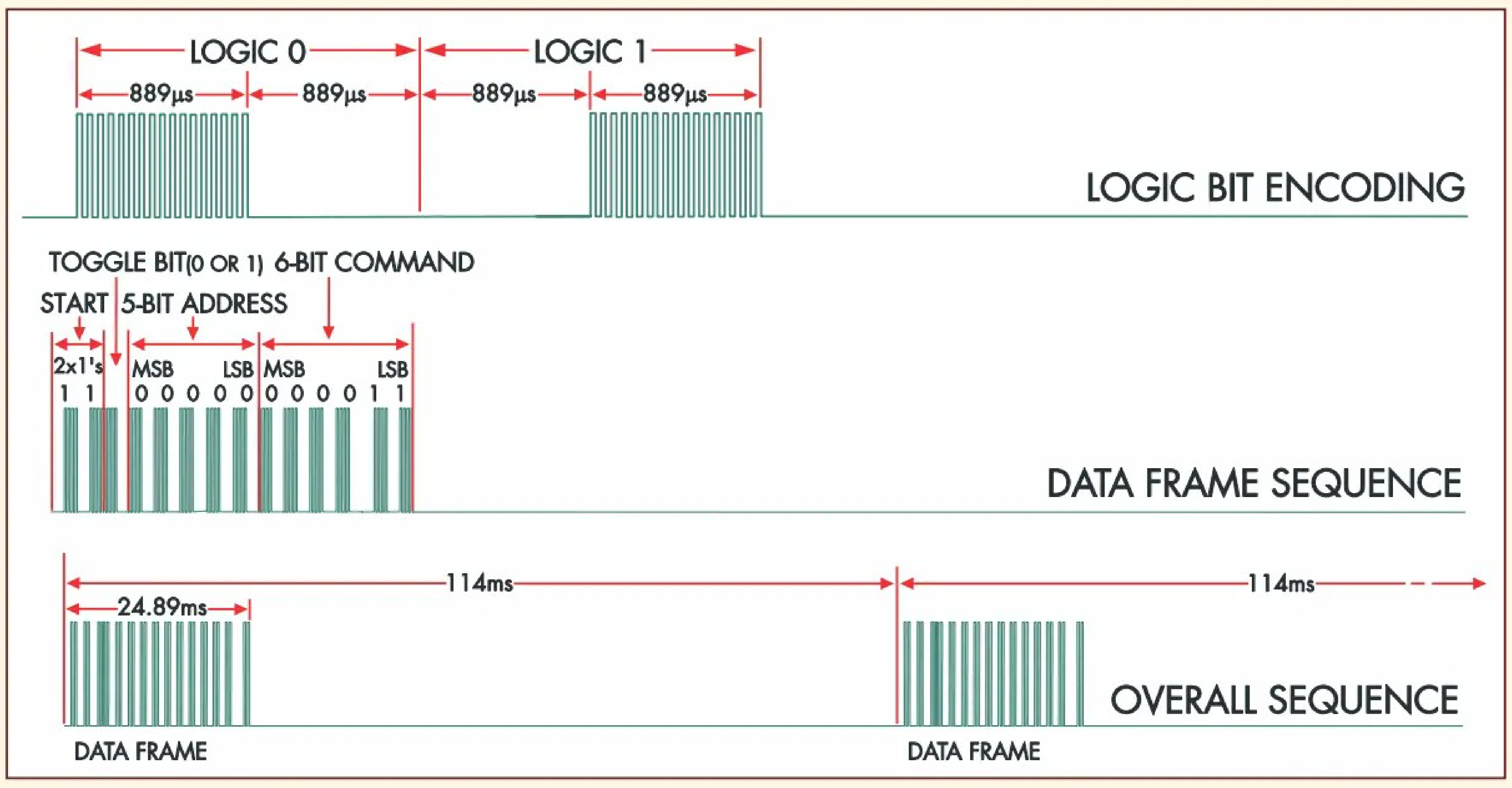
There is also the Pulse Width Protocol used by Sony and Pulse Distance Protocol, originating from NEC. For more details, see application note AN3053 by Freescale Semiconductors (formerly Motorola): **https://pemag.com.au/link/aapv**

Philips RC5 (Manchester-encoded) (36kHz)

For this protocol, the 0s and 1s are transmitted using 889μs bursts and pauses at 36kHz. A ‘1’ is an 889μs pause then an 889μs burst, while a ‘0’ is an 889μs burst followed by an 889μs pause. The entire data frame has start bits comprising two 1s followed by a toggle bit that could be a 1 or 0. More about the toggle bit later. The data comprises a 5-bit address followed by a 6-bit command. The most significant address and command bits come first.

When a button is held down, the entire sequence is repeated at 114ms intervals. Each repeat frame is identical to the first. However, if transmission stops, then the same button is pressed again, the toggle bit changes. This informs the receiver as to how long the button has been held down.

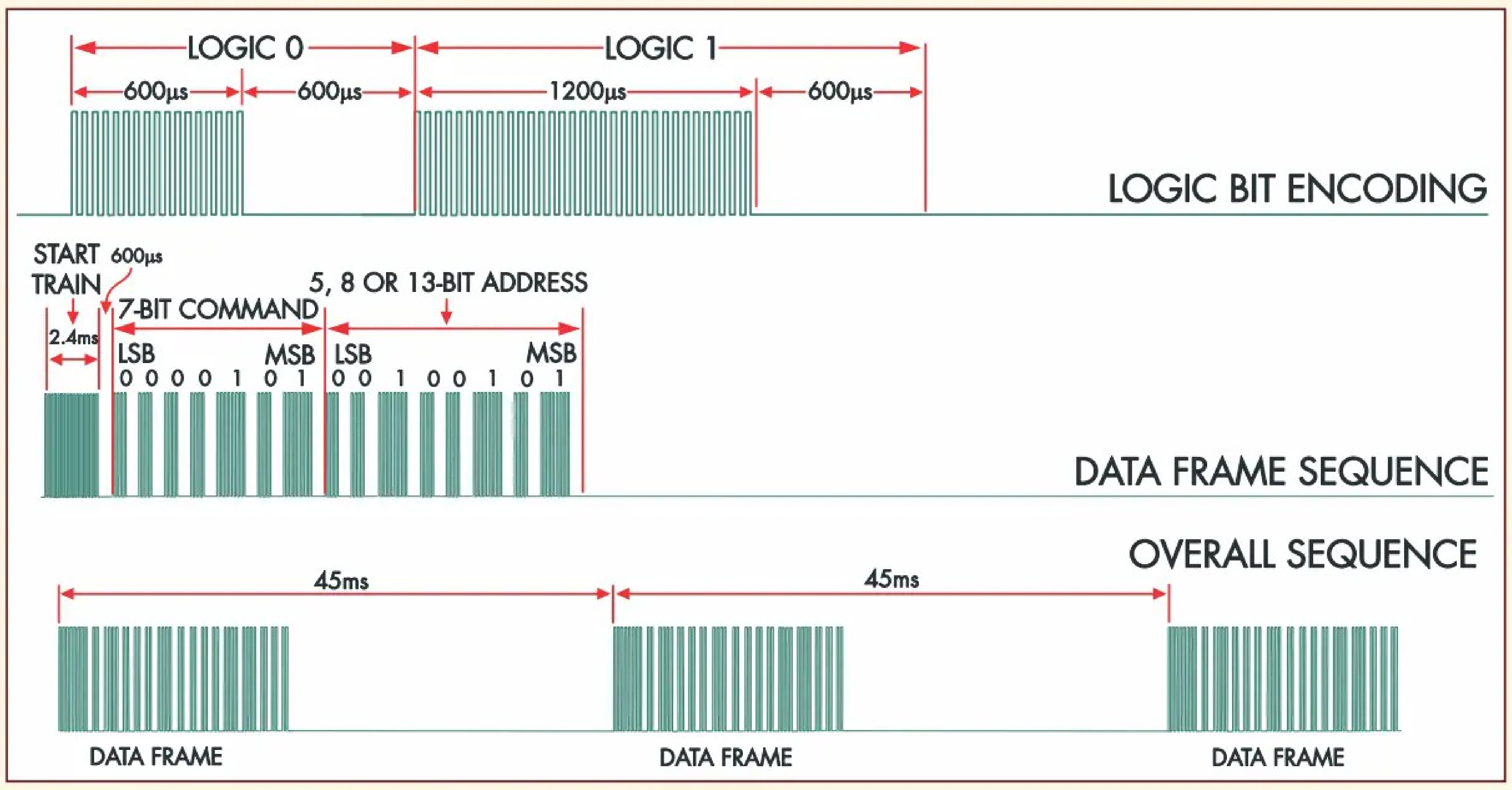
That’s so it can, for example, know when to increase volume at a faster rate after the button has been held down for some time.



Sony Pulse Width Protocol (40kHz)

This is also known also as SIRC, which is presumably an acronym for Sony Infra Red Code. For this protocol, the 0s and 1s are transmitted with a differing overall length. The pause period is the same at 600μs, but a ‘1’ is sent as a 1200μs burst at 40kHz, followed by a 600μs pause, while a ‘0’ is sent as a 600μs burst at 40kHz followed by a 600μs pause.

The entire data frame starts with a 2.4ms burst followed by a 600μs pause. The 7-bit command is then sent with the least significant bits first. The address bits follow, again with least significant bits first. The address can be five bits, eight bits or 13 bits long to make up a total of 12, 15 or 20 bits of data. Repeat frames are the entire above sequence sent at 45ms intervals.



NEC Pulse Distance Protocol (PDP) (38kHz)

For the NEC infrared remote control protocol, binary bits zero and one both start with a 560μs burst modulated at 38kHz. A logic 1 is followed by a 1690μs pause while a logic 0 has a shorter 560μs pause. The entire signal starts with a 9ms burst and a 4.5ms pause.

The data comprises the address bits and command bits. The address identifies the equipment type that the code works with, while the command identifies the button on the remote control which was pressed.

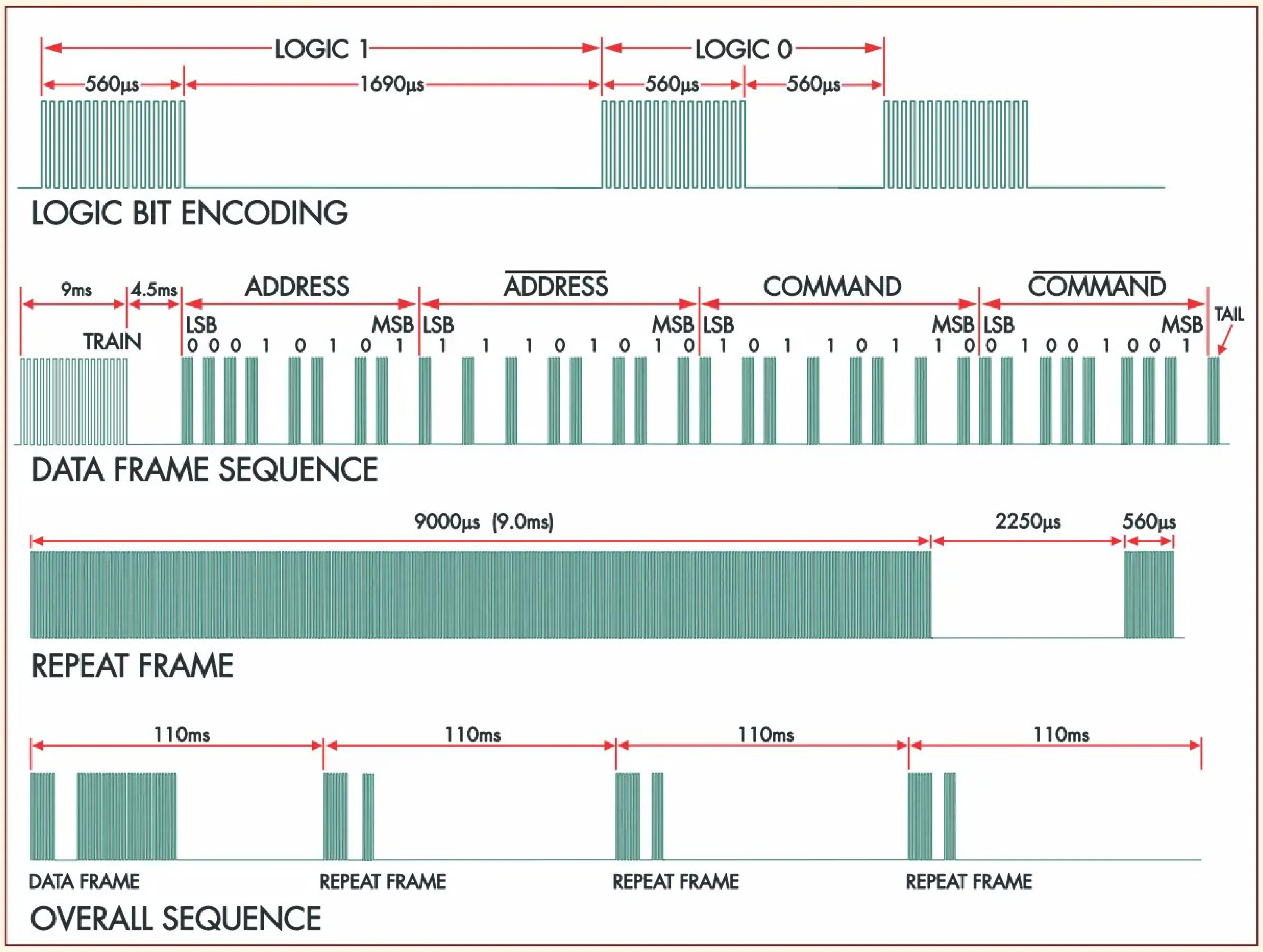
The second panel shows the structure of a single transmission. It starts with a 9ms burst and a 4.5ms pause. This is then followed by eight address bits and another eight bits which are the “one’s complement" of those same eight address bits (ie, the 0s become 1s and the 1s become 0s). An alternative version of this protocol uses the second series of eight bits for extra address bits.

The address signal is followed by eight command bits, plus their 1's complement, indicating which function (eg volume, source etc) should be activated. Then finally comes a 560µs "tail" burst to end the transmission. Note that the address and command data is sent with the least significant bit first.

The complementary command bytes are for detecting errors. If the complement data value received is not the complement of the data received then one or the other has been incorrectly detected or decoded. A lack of complementary data could also suggest that the transmitter is not using the PDP protocol.

After a button is pressed, if it continues to be held down, it will produce repeat frames. These consist of a 9ms burst, a 2.25ms pause and a 560µs burst. These are repeated at 110ms intervals.

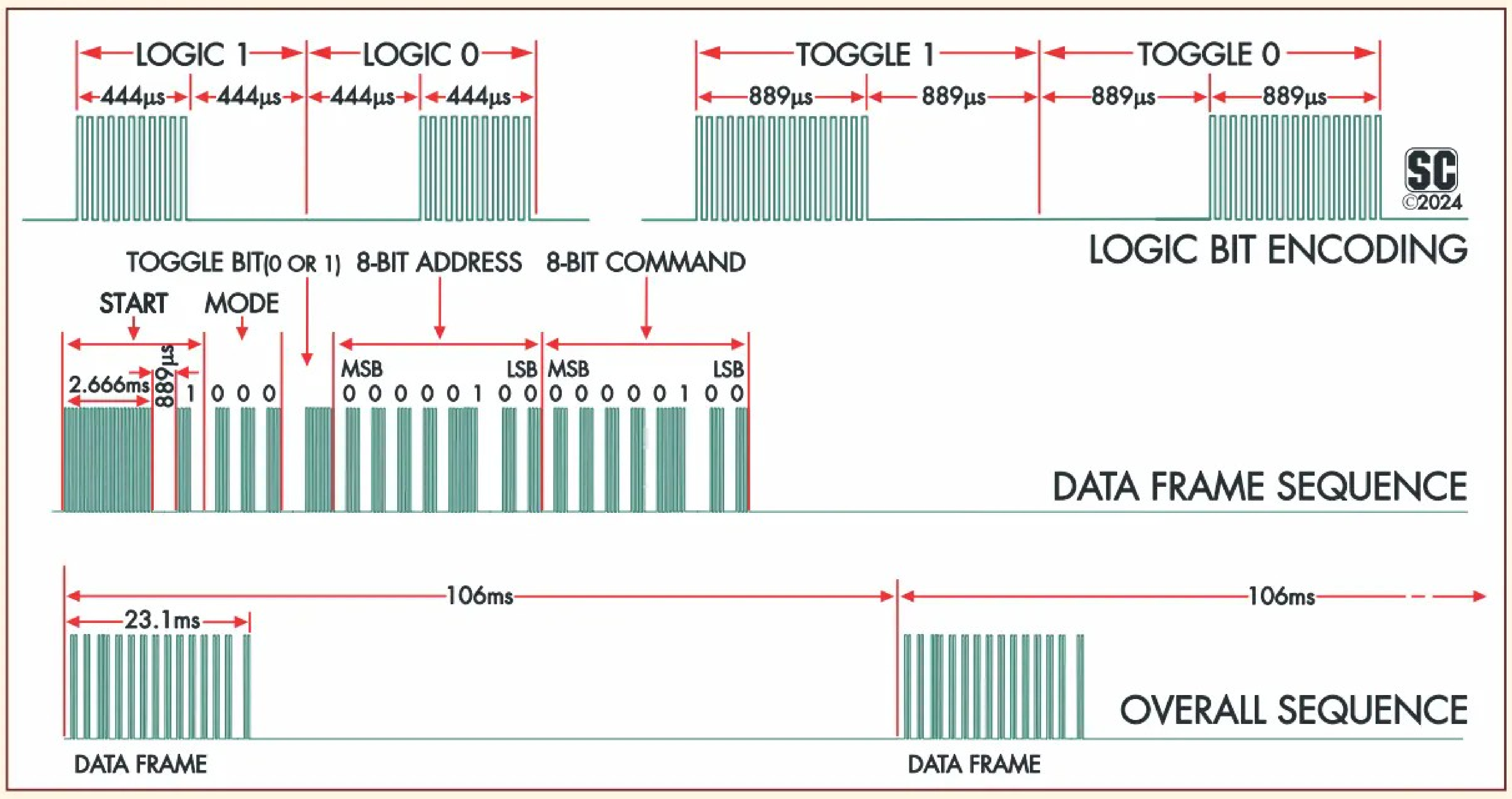
The repeat frame informs the receiver that it may repeat that particular function, depending on what it is. For example, volume up and volume down actions are repeated while the repeat frame signal is received but power off or mute would be processed once and not repeated with the repeat frame.


LOGIC 1
LOGIC 0
560µs
1690µs
560µs
560µs
LOGIC BIT ENCODING
9ms
4.5ms
TRAIN
ADDRESS
ADDRESS
COMMAND
COMMAND
LSB
MSB
0 0 0 1 0 1 0 1
LSB
MSB
1 1 1 0 1 0 1 0
LSB
MSB
1 0 1 1 0 1 1 0
LSB
MSB
0 1 0 0 1 0 0 1
TAIL
DATA FRAME SEQUENCE
9000µs (9.0ms)
2250µs
560µs
REPEAT FRAME
110ms
110ms
110ms
110ms
DATA FRAME
REPEAT FRAME
REPEAT FRAME
REPEAT FRAME
OVERALL SEQUENCE

Philips RC6 (Manchester-encoded) (36kHz)

0s and 1s are transmitted using 444µs bursts with 444µs pauses at 36kHz. The entire data frame has start bits comprising a 2.666ms burst followed by a pause for 889µs, then a '1' bit. After this, there is a 3-bit mode value, typically 000. The toggle bit comes after that; it uses an 889µs burst and 889µs pause instead of the 444µs used for the Mode, Address and Command bits.

The data is an 8-bit address followed by an 8-bit command, with the most significant bits first. The same sequence is repeated at 106ms intervals when a button is held down. If transmission stops and the same button is pressed again, the toggle bit changes state. This lets the receiver determine how long the button was held down.


LOGIC 1
LOGIC 0
444µs
444µs
444µs
444µs
TOGGLE 1
TOGGLE 0
889µs
889µs
889µs
889µs
SC ©2024
LOGIC BIT ENCODING
TOGGLE BIT (0 OR 1)
8-BIT ADDRESS
8-BIT COMMAND
START
MODE
2.666ms
889µs
1 0 0 0
MSB
LSB
0 0 0 0 0 1 0 0
MSB
LSB
0 0 0 0 0 1 0 0
DATA FRAME SEQUENCE
106ms
106ms
23.1ms
OVERALL SEQUENCE
DATA FRAME
DATA FRAME

Codes learned are stored in non-volatile flash memory. This ensures that the IR codes and other settings like momentary/toggle and the timer period are not lost if the power is cycled. All outputs are initially off when power is applied to the Receiver.

The 8-Channel Learning IR Remote Receiver fits neatly into a compact instrument enclosure. An acknowledge (ACK) LED and the eight channel status LEDs are mounted on the front, while the power input and channel output connections are at the rear. A 12V DC plugpack or similar supply powers the Receiver.

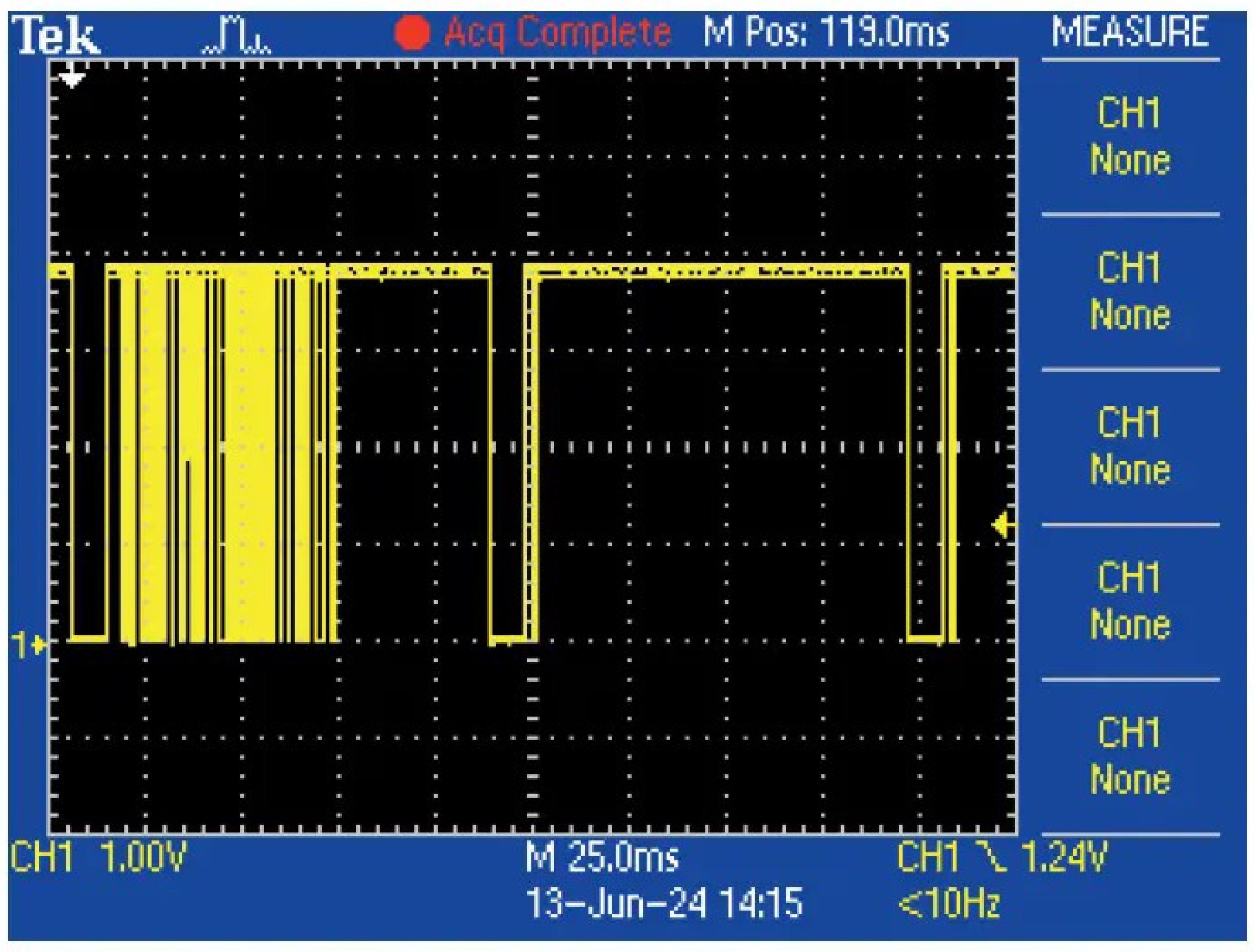
Circuit details

Referring to the circuit diagram, Fig.5, an infrared receiver (IRD1), sends signals to a PIC16F1459 microcontroller (IC1), which drives reed relays, NPN transistors or a combination of both, depending on how you configure the PCB.

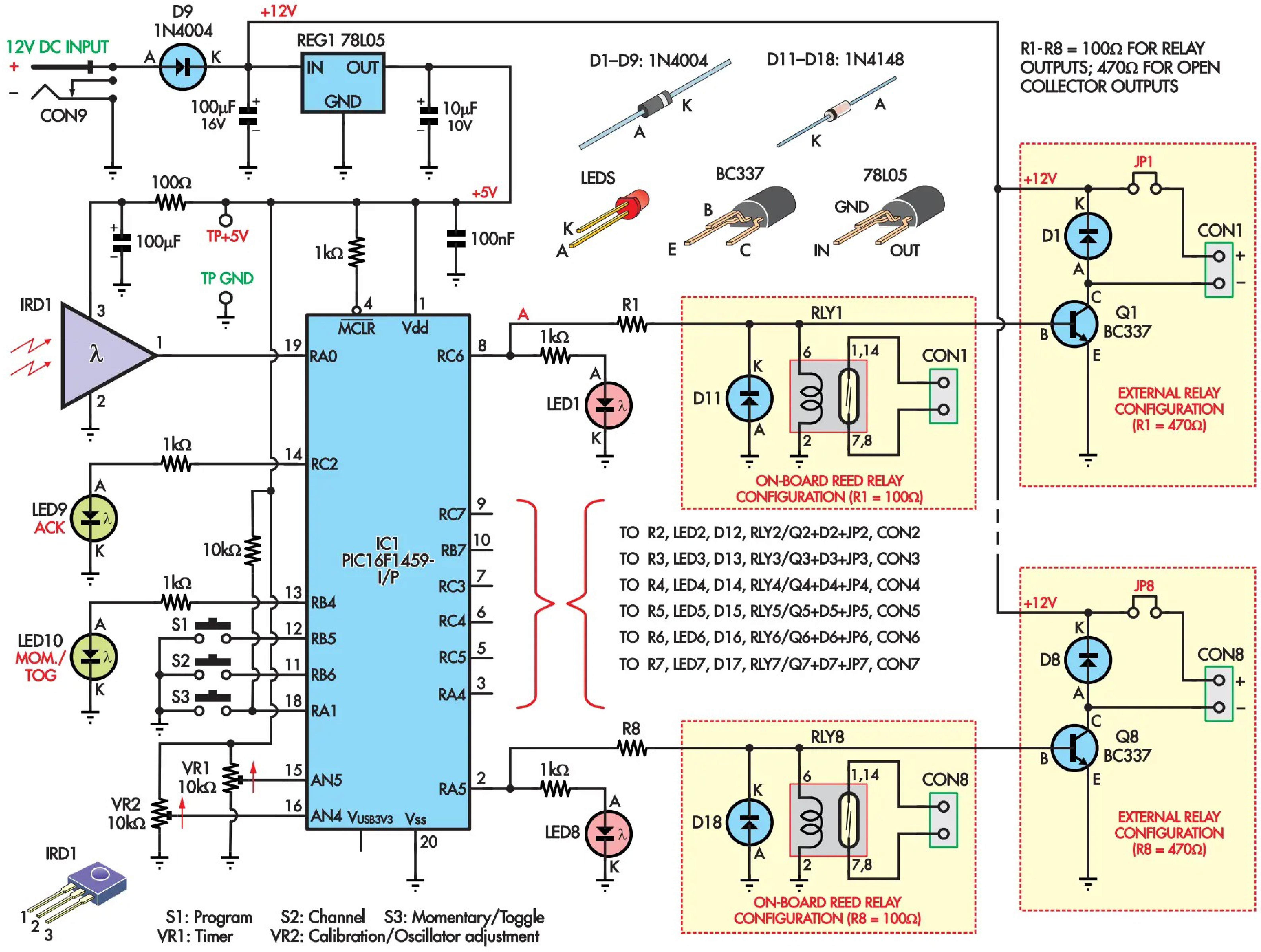
IRD1 includes an infrared detector, amplifier, bandpass filter (typically centred around 38kHz) and an automatic gain control (AGC). IRD1's output is normally high (5V) but goes low (near 0V) when it receives a 38kHz IR signal. This means that the infrared receiver removes the 38kHz modulation, with the output staying low for the duration of the frequency burst.

The supply for IRD1 is derived via a 100Ω resistor from the 5V rail and it is decoupled by a 100µF electrolytic capacitor. This is to keep electrical noise out of the supply for IRD1; it requires a steady supply as it contains a sensitive, high-gain amplifier.

The infrared signal is modulated so that the detector will ignore other infrared sources, such as halogen lamps, bar radiators and the sun. Bar


Tek
Acq Complete
M Pos: 119.0ms
MEASURE
CH1
None
CH1
None
CH1
None
CH1
None
CH1
None
1
CH1 1.00V
M 25.0ms
13-Jun-24 14:15
CH1 ╲ 1.24V
<10Hz

Scope 5: the repeat code sent by an NEC-style remote control when you hold down a button.


12V DC INPUT
+
–
CON9
D9
1N4004
A
K
+12V
REG1 78L05
IN
OUT
GND
100µF
16V
10µF
10V
100Ω
100µF
TP+5V
TP GND
1kΩ
100nF
+5V
IRD1
3
1
2
λ
D1–D9: 1N4004
D11–D18: 1N4148
LEDS
BC337
78L05
B
E
C
GND
IN
OUT
R1-R8 = 100Ω FOR RELAY OUTPUTS; 470Ω FOR OPEN COLLECTOR OUTPUTS
4
MCLR
Vdd
19
RA0
RC6
8
R1
1kΩ
LED1
RLY1
D11
6
1,14
2
7,8
CON1
ON-BOARD REED RELAY CONFIGURATION (R1 = 100Ω)
+12V
JP1
D1
Q1
BC337
EXTERNAL RELAY CONFIGURATION (R1 = 470Ω)
14
RC2
LED9
ACK
10kΩ
IC1
PIC16F1459-I/P
RC7
9
RB7
10
RC3
7
RC4
6
RC5
5
RA4
3
TO R2, LED2, D12, RLY2/Q2+D2+JP2, CON2
TO R3, LED3, D13, RLY3/Q3+D3+JP3, CON3
TO R4, LED4, D14, RLY4/Q4+D4+JP4, CON4
TO R5, LED5, D15, RLY5/Q5+D5+JP5, CON5
TO R6, LED6, D16, RLY6/Q6+D6+JP6, CON6
TO R7, LED7, D17, RLY7/Q7+D7+JP7, CON7
13
RB4
LED10
MOM./TOG
S1
S2
S3
12
RB5
11
RB6
18
RA1
VR1
10kΩ
VR2
10kΩ
15
AN5
16
AN4
VUSB3V3
Vss
20
RA5
2
R8
LED8
RLY8
D18
CON8
ON-BOARD REED RELAY CONFIGURATION (R8 = 100Ω)
JP8
D8
Q8
BC337
EXTERNAL RELAY CONFIGURATION (R8 = 470Ω)
IRD1
1
2
3
S1: Program
S2: Channel
S3: Momentary/Toggle
VR1: Timer
VR2: Calibration/Oscillator adjustment

SC ©2015 **8–CHANNEL IR LEARNING REMOTE RECEIVER**

Fig.5: the main part of the circuit comprises microcontroller IC1, infrared receiver IRD1, a few LEDs and pushbuttons and a simple linear power supply. While there are eight output sections, only two are shown; the other six are identical. Each section can either have a reed relay (as shown in the boxes in the middle) or a transistor and diode (as shown on the right).

radiators and halogen lamps produce a modulated signal at 100Hz (for 50Hz mains), while the sun produces a constant level of infrared that can vary slowly over time. These are all removed by the bandpass filter within IRD1.

Many general-purpose IR detectors centre the filter at 38kHz, allowing a frequency range from 36kHz to 40kHz to be received without too much attenuation from the bandpass filter. There may be a small amount of attenuation that reduces the reception range slightly, but not to any significant extent. RC5 and RC6 encodings use 36kHz modulation, NEC uses 38kHz and Sony uses 40kHz.

These varying frequencies mean we have to compromise with the infrared detector for it to work with all these protocols, with 38kHz being the best bet as it's in the middle of the range.

IRD1's output goes to the RA0 digital input of microcontroller IC1 (pin 19), which decodes the demodulated signal pulses and drives the outputs according to the infrared code sent by the handheld remote. Each output channel includes an indicator LED, driven via a 1kΩ resistor, and either a 100Ω resistor to drive a reed relay or a 470Ω resistor going to the base of an NPN transistor.

If a reed relay is used, a reverse-biased diode (D11-D18) clamps the back-EMF voltage from the relay's coil as it switches off. If an output transistor is used instead, a diode (D1-D8) clamps the back-EMF produced by any external relay coil it might be driving. Whenever the transistor is turned on, the external relay will be on.

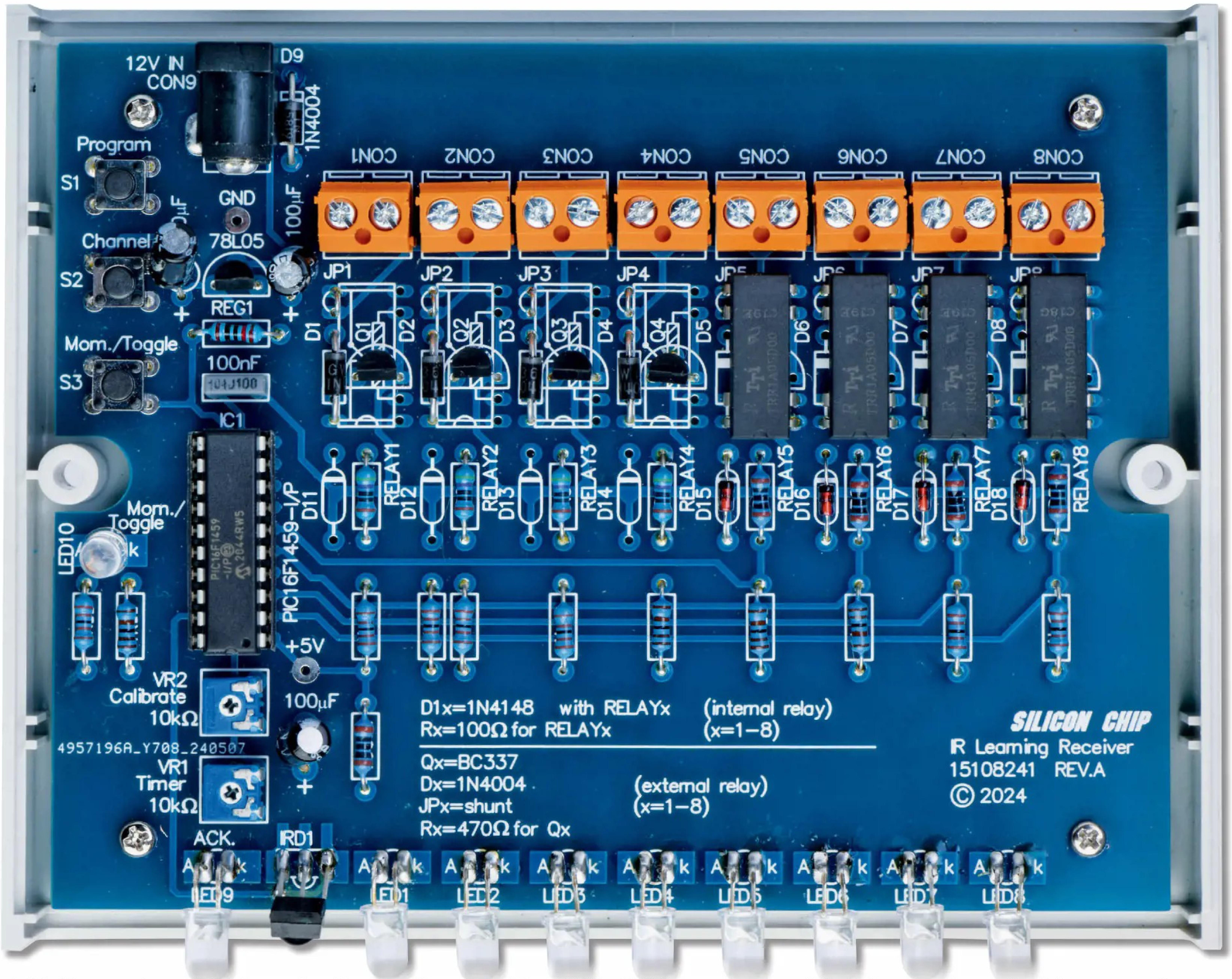
The circuit shows one output driven by the RC6 digital output (pin 8) and one driven by the RA5 digital output (pin 2), but six other outputs are also available, for a total of up to eight.

Any output configured as an open-collector type provides a +12V terminal suitable for driving an external 12V DC coil relay. This comes from the power input socket (CON9) via reverse-polarity protection diode D9.

The acknowledge (ACK) LED, LED9, is driven from IC1's RC2 digital output and flashes whenever an infrared signal is received. LED9 doubles up as a power indicator by glowing at about 6% brightness when an IR signal is not being received. The ACK LED also provides indications during the process of learning infrared codes; more on that later.

Pushbutton switches S1, S2 and S3, connected to IC1's RB5, RB6 and RA1 digital inputs, are used during the learning process. Those three inputs are held high (at +5V) unless pulled to

8-channel Learning IR Remote


12V IN
CON9
D9
1N4004
Program
S1
GND
100µF
Channel
78L05
S2
REG1
Mom./Toggle
100nF
S3
IC1
CON1
CON2
CON3
CON4
CON5
CON6
CON7
CON8
JP1
JP2
JP3
JP4
D1
D2
D3
D4
D5
D6
D7
D8
Q1
Q2
Q3
Q4
RELAY1
RELAY2
RELAY3
RELAY4
RELAY5
RELAY6
RELAY7
RELAY8
D11
D12
D13
D14
D15
D16
D17
D18
LED10
Mom./Toggle
PIC16F1459–I/P
+5V
VR2
Calibrate
10kΩ
100µF
4957196A_Y708_240507
VR1
Timer
10kΩ
D1x=1N4148 with RELAYx
Rx=100Ω for RELAYx
(internal relay)
(x=1–8)
Qx=BC337
Dx=1N4004
JPx=shunt
Rx=470Ω for Qx
(external relay)
(x=1–8)
SILICON CHIP
IR Learning Receiver
15108241 REV.A
© 2024
ACK.
IRD1
LED9
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8

A bird's eye view of the Learning Remote Receiver. The CON1-CON4 outputs are driven by transistors in this case, and CON5-CON8 by relays. The board allows either style to be used to drive any of the eight outputs.

0V when the corresponding button is pressed. The RA1 input is pulled high via a 10kΩ resistor to the 5V supply, while the RB5 and RB6 inputs are held high by pullup currents provided internally by IC1.

Trimpots VR1 and VR2 provide adjustments for the timer and IC1's oscillator. These connect to the AN5 and AN4 analog inputs, and IC1 converts the voltage at the wiper of each trimpot to a digital value. VR1 allows the timer for each channel to be adjusted from ⅛th of a second to 32 seconds.

Frequency adjustment

VR2 allows IC1's internal oscillator to be trimmed. Typically, it is set to its mid position so IC1's internal oscillator runs at the factory calibration rate (usually within 3% of nominal at 25°C). This oscillator is used as the time base for decoding the IR codes. Having an accurate time base provides reliable IR code detection.

While handheld IR remotes should transmit according to timing specifications, the timing can vary between remotes because many use a relatively inaccurate ceramic resonator for timing. These are used since they are cheaper than crystals and also smaller. The accuracy for low-cost versions is typically ±5%.

While IC1's decoding of IR signals does have some tolerance, having the adjustment allows for extra variation. VR2 can be adjusted to accommodate variations in IC1's oscillator as well as the IR remote control's. It allows IC1's frequency to be adjusted by ±6% in 128 steps.

The 5V supply for IRD1 and IC1 comes from REG1, a 78L05 regulator. A 100µF electrolytic capacitor bypasses its input, while a 10µF capacitor filters its output. IC1's supply is also bypassed by a 100nF capacitor close to its supply pins.

Construction

All parts are installed on a PCB coded 15108241 that measures 130 × 101.5mm. This can be housed in a 140 × 110 × 35mm plastic case, with optional panel labels affixed to the front and rear panels.

Fig.6 shows the layout of the parts on the PCB with all eight reed relays fitted. In contrast, Fig.7 shows the identical layout but with open-collector transistor outputs suitable for driving external 12V DC relays or other 12V loads. You can mix and match the two output types, and you don't have to populate all eight outputs.

As shown in the photos of our prototype, we installed open-collector

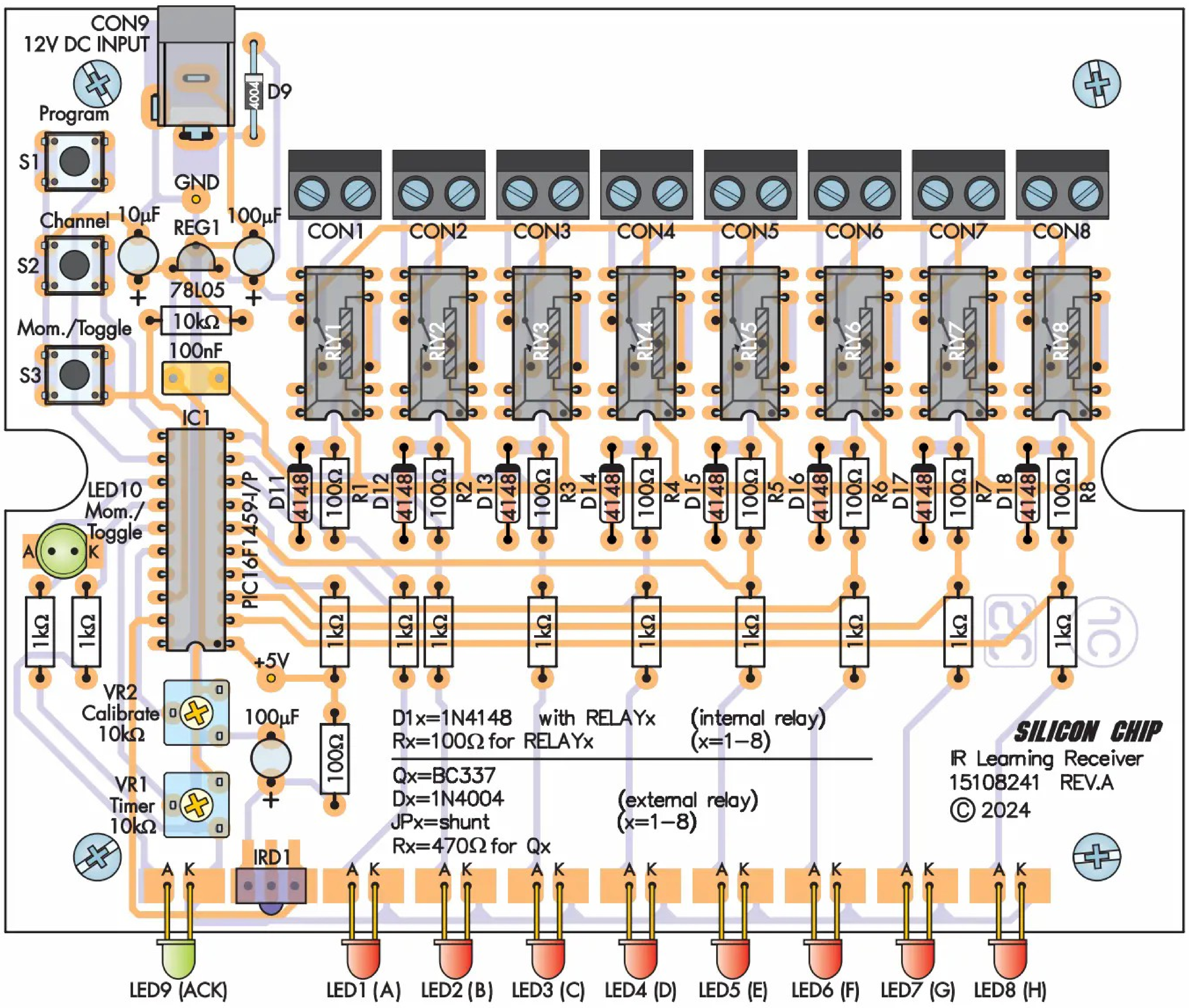

CON9
12V DC INPUT
D9
Program
S1
GND
Channel
10µF
REG1
100µF
S2
78L05
10kΩ
Mom./Toggle
100nF
S3
IC1
CON1
CON2
CON3
CON4
CON5
CON6
CON7
CON8
RLY1
RLY2
RLY3
RLY4
RLY5
RLY6
RLY7
RLY8
D11
D12
D13
D14
D15
D16
D17
D18
4148
100Ω
R1
R2
R3
R4
R5
R6
R7
R8
LED10
Mom./
Toggle
A
K
PIC16F1459-I/P
1kΩ
+5V
VR2
Calibrate
10kΩ
VR1
Timer
10kΩ
IRD1
D1x=1N4148 with RELAYx (internal relay)
Rx=100Ω for RELAYx (x=1–8)
Qx=BC337
Dx=1N4004
JPx=shunt
Rx=470Ω for Qx
(external relay)
(x=1–8)
SILICON CHIP
IR Learning Receiver
15108241 REV.A
© 2024
LED9 (ACK)
LED1 (A)
LED2 (B)
LED3 (C)
LED4 (D)
LED5 (E)
LED6 (F)
LED7 (G)
LED8 (H)

Fig.6: the PCB populated with eight reed relays. With these relays, the outputs are not polarised. You don't need to install all eight relays if you need fewer.

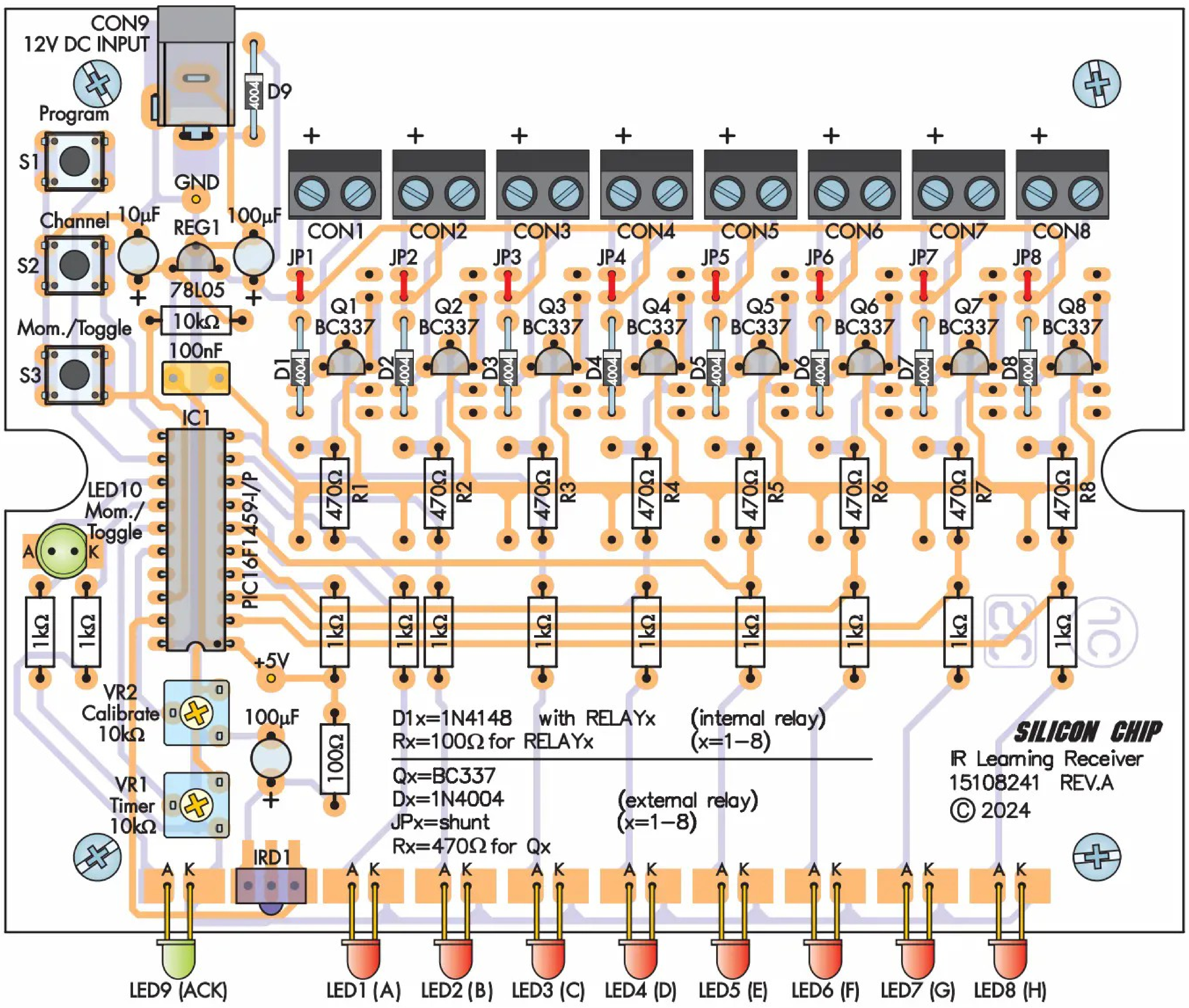

CON9
12V DC INPUT
D9
Program
S1
GND
Channel
10µF
REG1
100µF
S2
78L05
10kΩ
Mom./Toggle
100nF
S3
IC1
+
CON1
CON2
CON3
CON4
CON5
CON6
CON7
CON8
JP1
JP2
JP3
JP4
JP5
JP6
JP7
JP8
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
BC337
D1
D2
D3
D4
D5
D6
D7
D8
4004
470Ω
R1
R2
R3
R4
R5
R6
R7
R8
LED10
Mom./
Toggle
A
K
PIC16F1459-I/P
1kΩ
+5V
VR2
Calibrate
10kΩ
VR1
Timer
10kΩ
IRD1
D1x=1N4148 with RELAYx (internal relay)
Rx=100Ω for RELAYx (x=1–8)
Qx=BC337
Dx=1N4004
JPx=shunt
Rx=470Ω for Qx
(external relay)
(x=1–8)
SILICON CHIP
IR Learning Receiver
15108241 REV.A
© 2024
LED9 (ACK)
LED1 (A)
LED2 (B)
LED3 (C)
LED4 (D)
LED5 (E)
LED6 (F)
LED7 (G)
LED8 (H)

Fig.7: this is like Fig.6 but all eight output sections have been populated with transistors. They can drive external loads directly or be used to control external relays. You can also mix and match relays and transistors. The wire links feed 12V to the left side of the terminals (marked +).

transistor outputs for the first four channels and relays for the last four channels.

Regardless of whether you populate all eight outputs, you should fit LED1 to LED8 and their associated 1kΩ resistors. As well as showing activated channels, they display the selected timeout period during the learning procedure.

Begin assembly by fitting the resistors. The parts list shows the resistor colour codes, but you should also check their values using a DMM before soldering them to the PCB. Be sure to fit the correct values for resistors R1-R8: 100Ω for reed relays or 470Ω for open-collector transistor outputs. Keep the lead off-cuts, as you may need them later.

The diodes can go in next. D11-D18 are 1N4148 types, while D1-D9 are 1N4004s. Take care that the diodes are all orientated correctly.

Next, install the 20-pin DIL socket for IC1 (notched end to the lower edge of the PCB). The capacitors can then be soldered in place, ensuring that the three electrolytics are orientated correctly. The 100nF capacitor can be fitted either way around.

Follow by installing the DC socket (CON9) and switches S1, S2 and S3. After that, fit transistors Q1-Q8 and/or relays RLY1-RLY8 with the notched ends downwards. Be sure to place REG1 (78L05) in the correct position. It has the same TO-92 body as the transistors.

Jumper wires JP1-JP8 can now be installed in any channels where a transistor is fitted. These only need to be very short (less than 5mm) and can be fashioned from resistor lead off-cuts bent in a 'U' shape.

Trimpots VR1 & VR2 can be installed now, along with screw terminals CON1-CON8. Ensure that the terminals sit flush against the PCB and that their wire entry holes are toward the board's top edge before soldering their pins.

LEDs & infrared detector

LED10 can be installed with its body a millimetre or two above the PCB. Be sure to install it with the correct polarity: the longer anode lead goes to the left, as indicated on the overlay diagrams. Mount the remainder of the LEDs, as shown in Fig.8.

Their leads must be bent down by 90° 6mm from their bodies. That's best done using a 6mm-wide card-

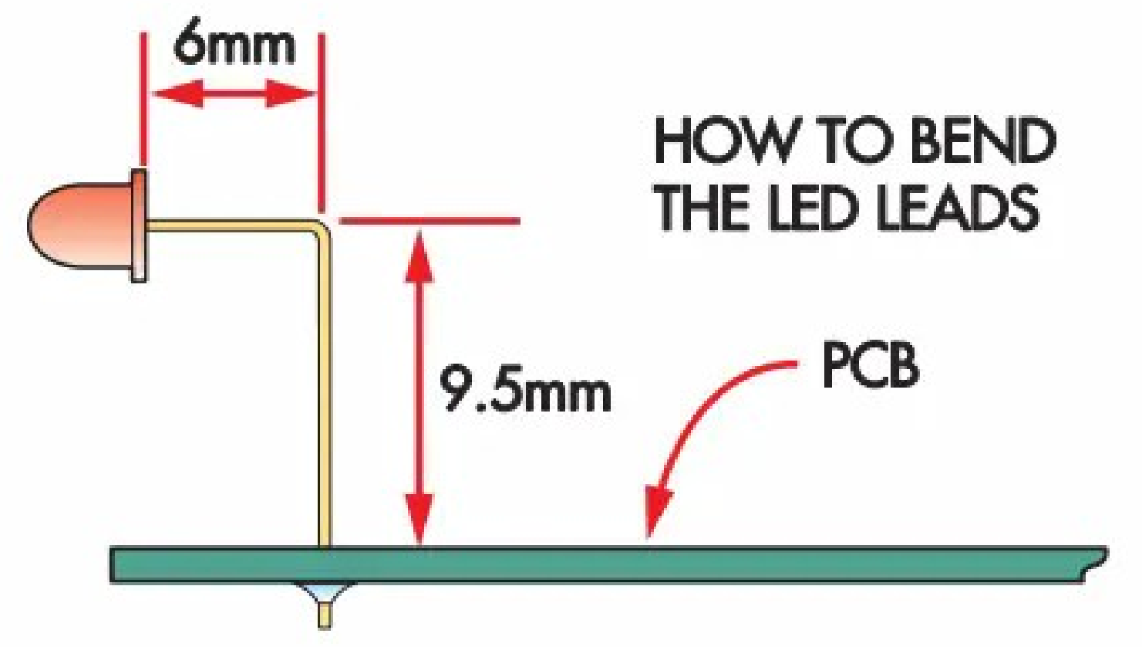

6mm
HOW TO BEND THE LED LEADS
9.5mm
PCB

Fig.8: bend LED1-LED9 like this so they will reach the holes in the front panel. Make sure you bend the leads in the right direction so that the longer (anode) leads will be on the left when mounted on the PCB, as shown in Figs.6 & 7.

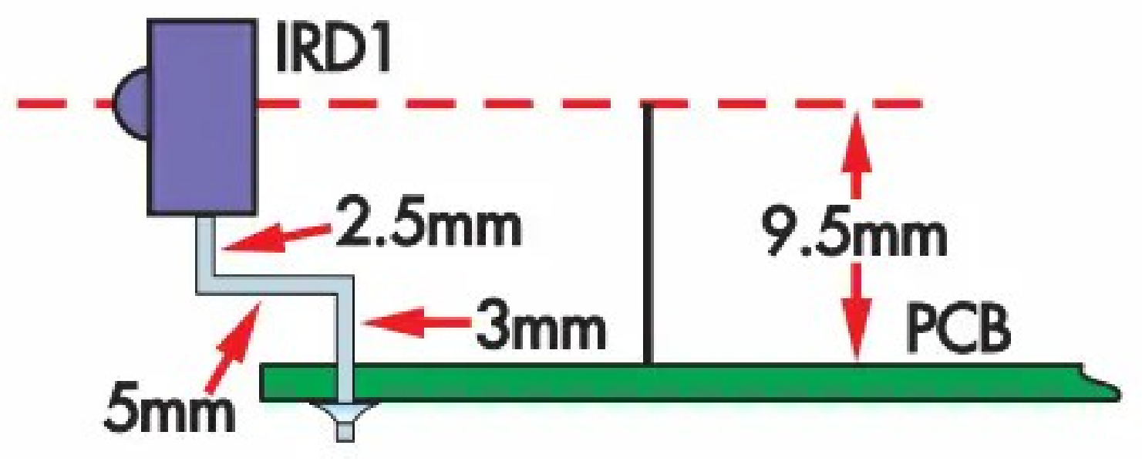
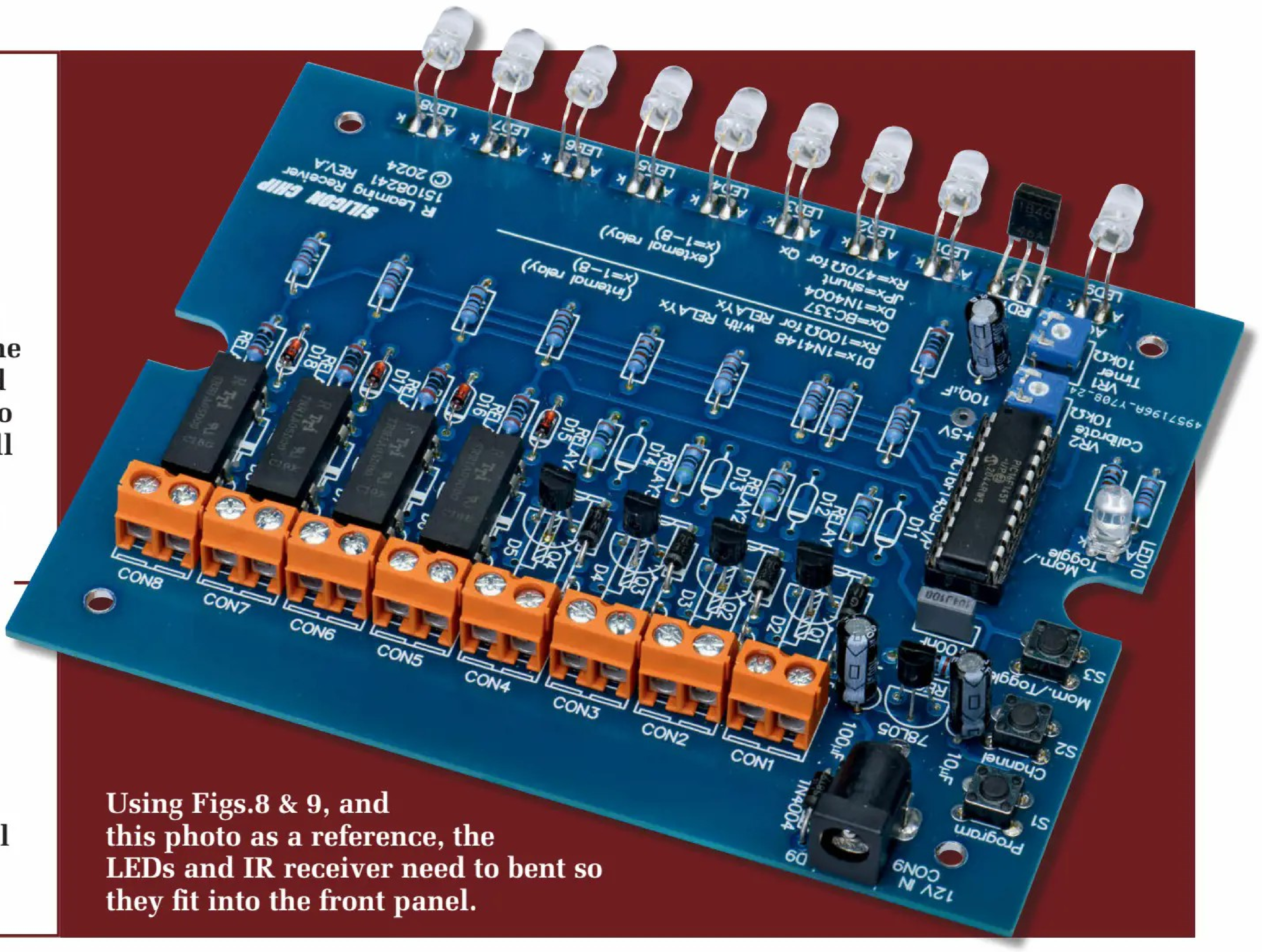

IRD1
2.5mm
9.5mm
3mm
PCB
5mm

Fig.9: similarly, by bending the IR receiver leads like this, it will reach the associated hole in the front panel.



Using Figs.8 & 9, and this photo as a reference, the LEDs and IR receiver need to bent so they fit into the front panel.

board template. Make sure that each LED's cathode (K) lead (the shorter of the two) is towards you before bending it as shown. That way, the LEDs go in with the correct polarity, with the anode to the left-most hole in the PCB.

Don't solder the LEDs to the PCB at this stage. We'll do that later, with the PCB in the case.

Having prepared the LEDs, you can now bend the infrared detector's leads as shown in Fig.9. Solder it in place with the centre of its lens 9.5mm above the PCB.

Drilling the case

The next step is to drill the front and rear panels of the enclosure. The drilling template, Fig.10, shows where the holes are located and their sizes. You need to drill 6mm diameter holes for the nine LEDs and their bezels, as well as for the infrared receiver, IRD1. The holes in the rear panel are for cable glands and the DC socket.

We used two glands, but the total number can be increased if you can't fit all the output wiring through just two glands. Their holes should be at least 22mm apart in the region shown. The 12mm holes for the glands are best made using a small pilot drill to begin with, carefully enlarged to size using a tapered reamer.

Final assembly

Once all the holes have been drilled, the PCB can be placed into the case. The nine LEDs can then be adjusted by cutting the leads shorter if they hit the base of the case. Next, insert the LEDs into the front panel holes (without the LED bezels initially) and fit the PCB and front panel into the enclosure.

Check that each LED is correctly orientated and that it protrudes through its front panel hole before soldering its leads on the top of the PCB. Once they have all been soldered, remove the board and also solder them on the underside of the PCB, then trim the leads further.

Now check that the infrared detector's lens aligns correctly with its front-panel hole. If not, bend its leads until it's centred.

Testing

Apply power using a 12V DC plugpack and check that the voltage

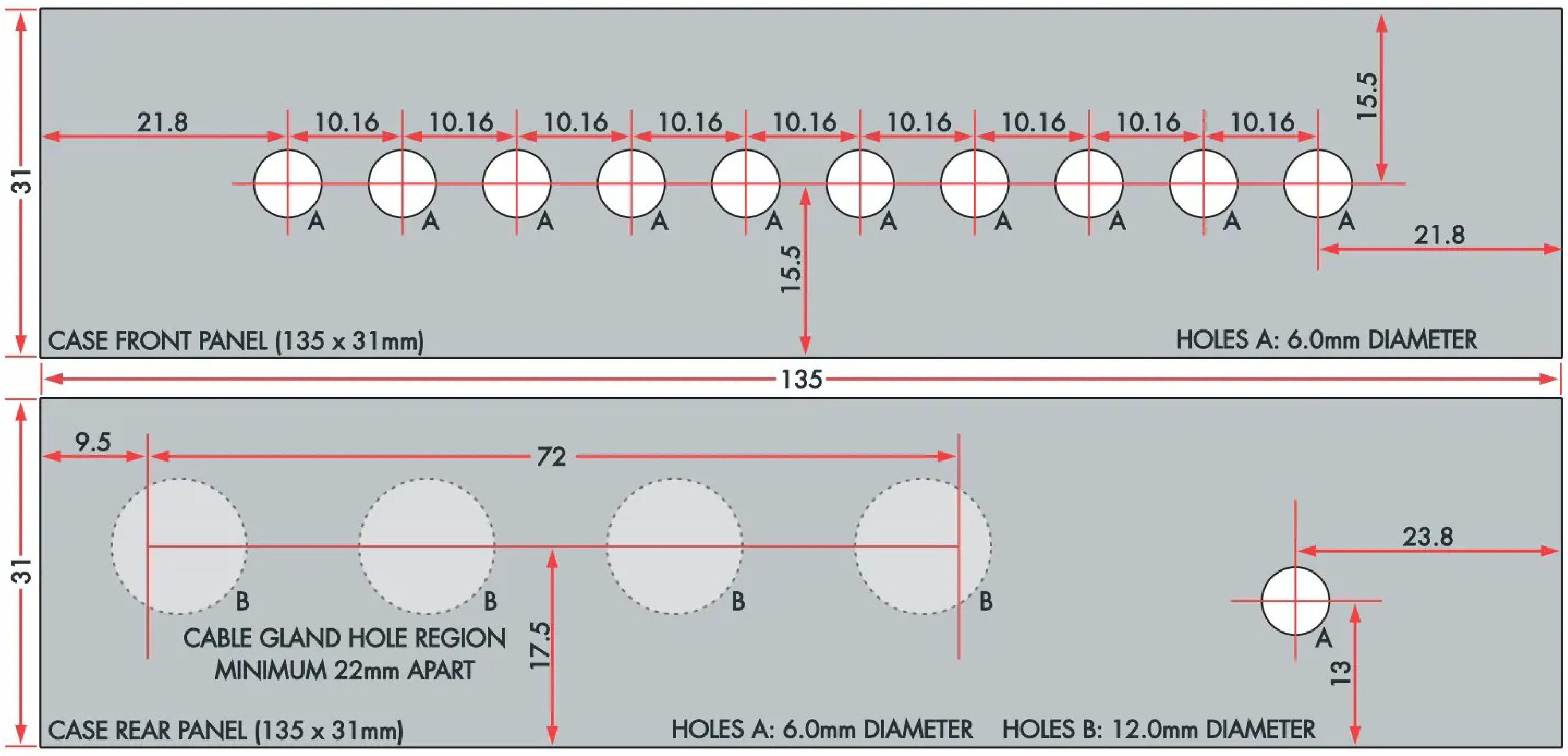

21.8
10.16
10.16
10.16
10.16
10.16
10.16
10.16
10.16
10.16
15.5
31
A
15.5
21.8
CASE FRONT PANEL (135 x 31mm)
HOLES A: 6.0mm DIAMETER
135
9.5
72
31
23.8
B
B
B
B
A
CABLE GLAND HOLE REGION MINIMUM 22mm APART
17.5
13
CASE REAR PANEL (135 x 31mm)
HOLES A: 6.0mm DIAMETER
HOLES B: 12.0mm DIAMETER

Fig.10: the front and rear panel drilling details. These diagrams can be printed/copied at actual size and used as templates. We drilled two 12mm holes for cable glands, but you can have up to four if needed. Ensure they're in the specified zone and a minimum of 22mm apart. *All dimensions are in millimetres.*

between pins 1 and 20 of IC1's socket is close to 5V (4.85-5.15V). If no voltage is present, check diode D9's polarity and the polarity of the 12V DC supply (the centre of the plug should be positive).

Also ensure that REG1 is correctly orientated and all leads have been correctly soldered to their PCB pads. If the supply checks out, switch off the power and install IC1, ensuring that its notched end faces toward the front and all its pins correctly go into the socket.

Set VR2 to its mid position. VR1 can be set fully anti-clockwise initially, for a 125ms timeout, so it is easier to check the momentary and toggle operations for the channel outputs.

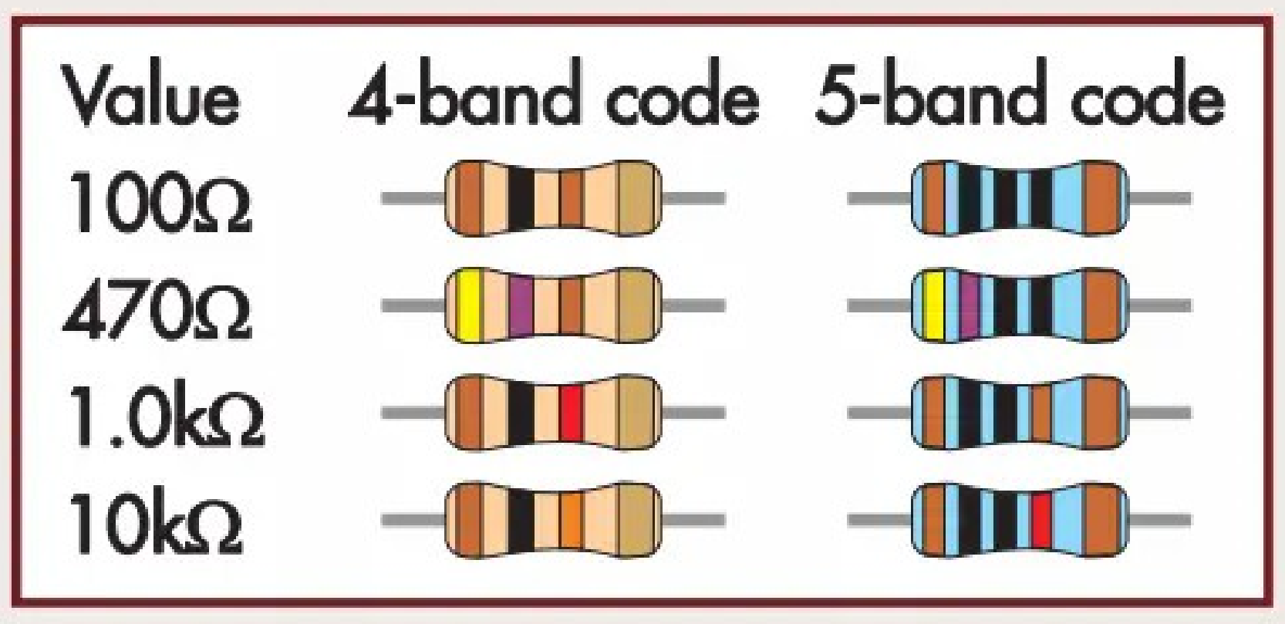
Parts List – 8-Channel IR Remote Receiver

1 double-sided PCB coded 15108241, 130 × 101.5mm
1 140 × 110 × 35mm plastic case **[Bud Industries PC-11480-LG, Gainta G738, Multicomp Pro G738; Farnell 1094711]**
2 panel labels, 131 × 28mm (**optional**)
1 12V DC plugpack rated at 150mA or more (see text)
3 SPST vertical tactile switches with ~0.7mm actuators (S1-S3)
8 2-way screw terminals, 5.08mm pitch (CON1-CON8; as required)
1 2.1mm or 2.5mm inner diameter PCB-mount DC socket to suit plugpack (CON9)
2 10kΩ mini top-adjust trimpots (VR1, VR2)
2 cable glands for 3-6.5mm cable
1 20-pin DIL IC socket
9 5mm LED bezels
4 No.4 self-tapping screws

Semiconductors
1 PIC16F1459-I/P microcontroller programmed with 1510824A.HEX (IC1)
1 TSOP4838 or similar 36-38kHz IR receiver (IRD1)
1 78L05 5V 100mA regulator (REG1)
8 high-brightness 5mm red LEDs (LED1-LED8)
2 high-brightness 5mm green LEDs or other colour (LED9, LED10)
1 1N4004 1A diode (D9)

Capacitors
2 100µF 16V PC electrolytic
1 10µF 16V PC electrolytic
1 100nF 50V MKT polyester or MLCC

Resistors *(all ¼W, 1% axial)*
1 10kΩ
11 1kΩ
1 100Ω



Value	4-band code	5-band code
100Ω		
470Ω		
1.0kΩ		
10kΩ		

Extra parts for reed relay outputs (per output, up to 8 total)
1 SPST DIP 5V reed relay (RLY1-RLY8)
1 1N4148 75V 200mA diode (D11-D18)
1 100Ω ¼W 1% axial resistor (R1-R8)

Extra parts for open-collector transistor outputs (per output, up to 8 total)
1 BC337 65V 100mA NPN transistor (Q1-Q8)
1 1N4004 1A diode (D1-D8)
1 470Ω ¼W 1% axial resistor (R1-R8)

Learning codes

The 8-Channel Learning IR Remote Receiver can learn infrared codes matching NEC, Sony, RC5 and RC6 protocols. These are commonly used in many handheld IR remote controls.

Each channel should be programmed using a different button on the handheld remote. You don't have to use the same remote to operate each channel. You can use different remote controls, provided they produce one of the supported protocols.

Once you start the learning mode, you have 20 seconds to finish this procedure before it times out and returns to the normal operating mode.

To program each channel, press the Program switch (S1). This will fully light the ACK LED on the front panel. One of the channel LEDs will also be lit, showing the currently selected channel.

Initially, this will be channel 1, but other channels can be selected by pressing the Channel switch (S2). Each press will choose the next channel; after 8, it will return to channel 1.

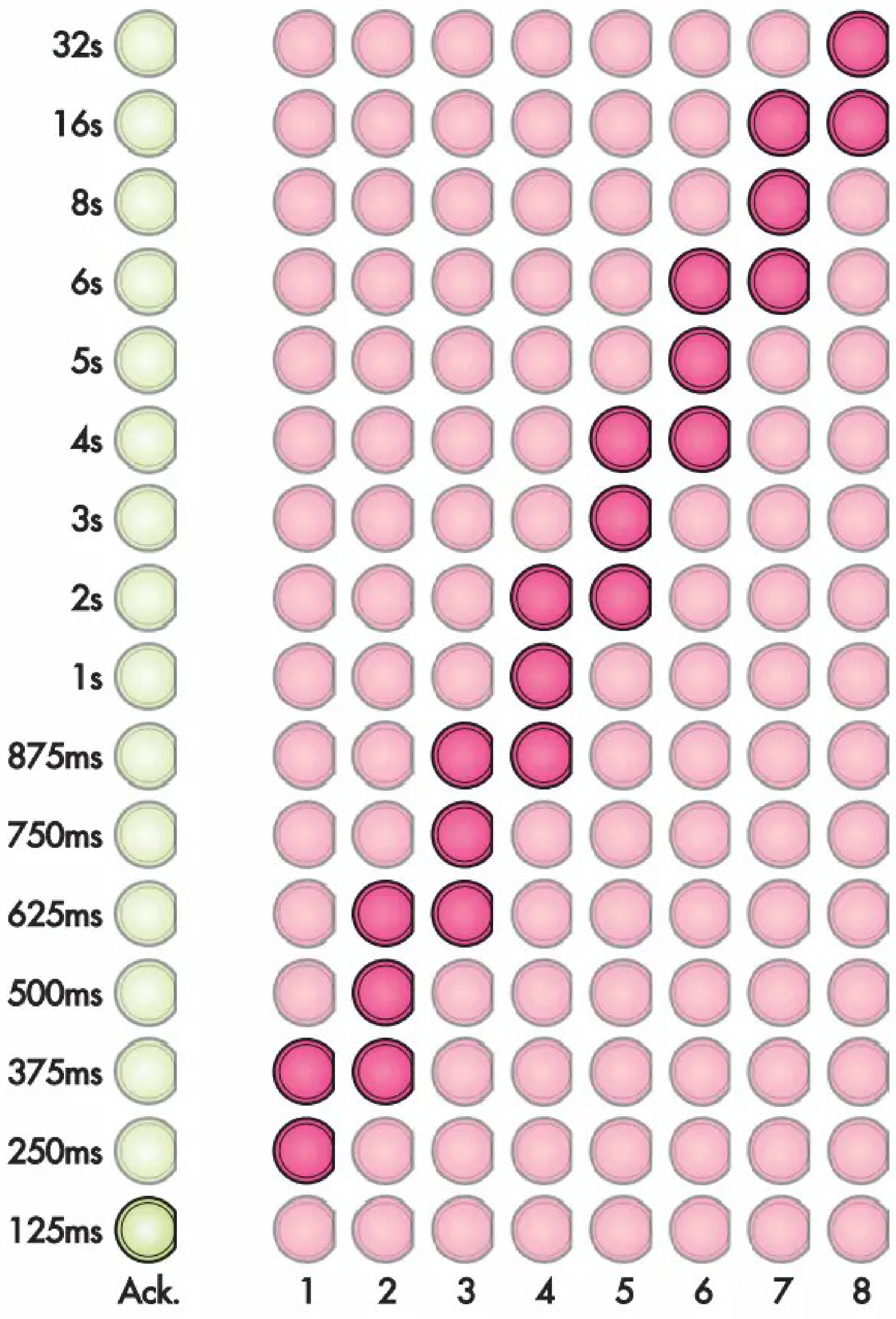
The Momentary/Toggle (MOM/TOG) LED will indicate the current selection for that channel. It lights for 125ms every second to show the momentary selection, or lights solid to show the toggle option is selected. Pressing S3 selects between momentary and toggle action.

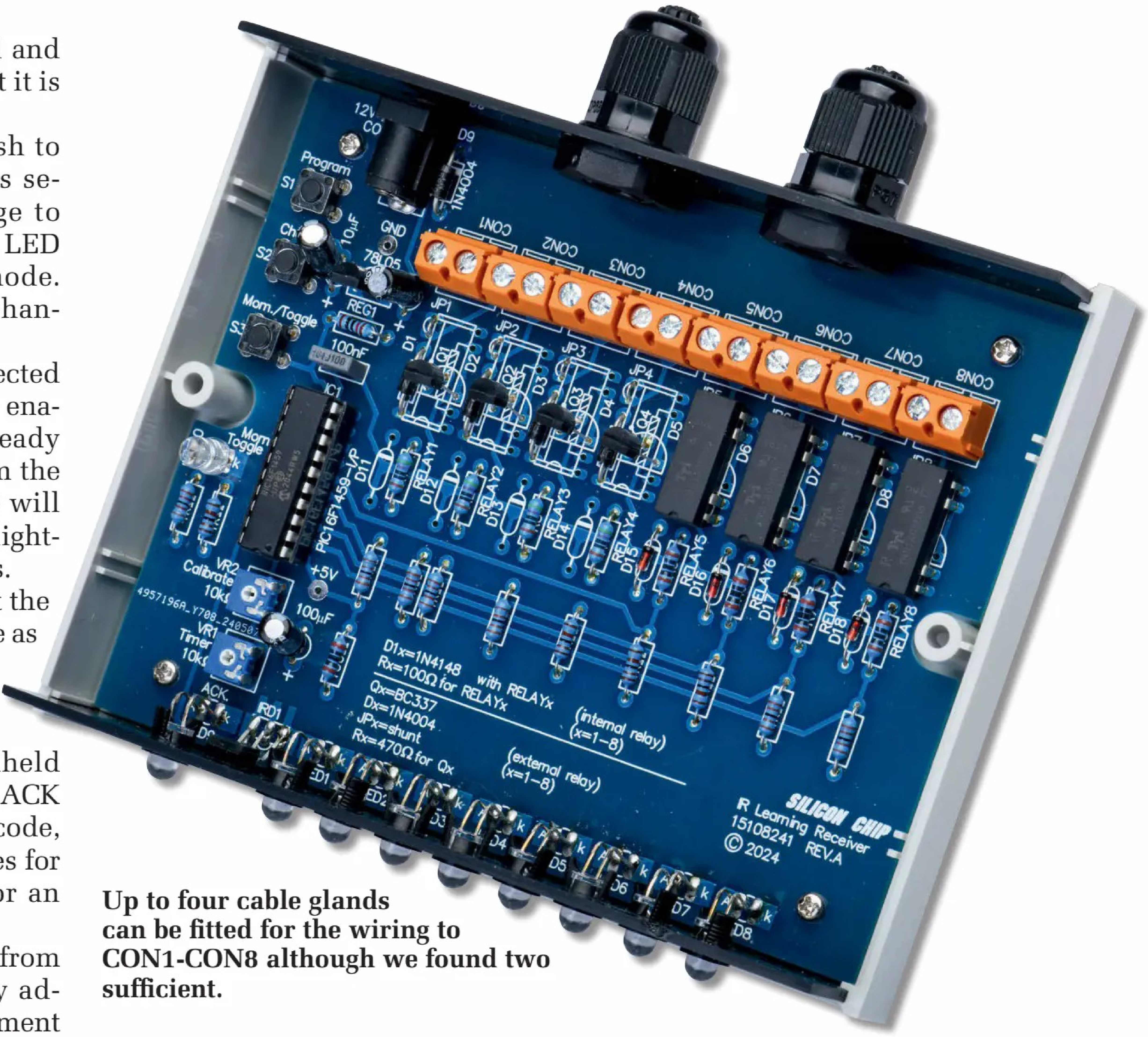
When momentary is selected, the time the channel is on (once programmed) is set by the timer. The timer value for the selected channel is adjusted using VR1. Timer values range from 125ms to 1s in eight 125ms steps, then options of two, three, four, five, six, eight, 16 and 32 seconds.

To set the timer, press and hold the MOM/TOG switch for at least 600ms. This will change the channel LEDs from showing the selected channel to displaying the chosen timer period instead.

If VR1 is fully anti-clockwise, none of the channel LEDs will light, but the ACK LED will be fully lit. For other timer periods, the ACK LED will be off, and the 8-channel LEDs will show the timer setting as per Fig.11, like a dot bargraph.

Adjust VR1 for the timer period required. When S3 is released, the channel display and ACK LED will return





Up to four cable glands can be fitted for the wiring to CON1-CON8 although we found two sufficient.

to showing the selected channel and fully lit ACK LED to indicate that it is still in the programming mode.

The MOM/TOG LED will flash to show that momentary action is selected. If you decide to change to toggle, press S3 again and the LED will stay lit, indicating toggle mode. In this case, the timer for that channel is inactive.

Once the channel has been selected and the timer adjusted (or toggle enabled), press S1. This makes it ready to receive an infrared signal from the handheld remote. The ACK LED will flash in readiness, with the LED lighting for 125ms every two seconds.

A lack of flashes indicates that the Receiver hasn't accepted the code as valid. It will flash at 1Hz with a 50% duty cycle. Point the handheld remote toward the receiver and press a button on the handheld remote. If the IR code is valid, the ACK LED will flash once for an NEC code, twice for a Sony code, three times for an RC5 code and four times for an RC6 code.

If you are sure that the code from the remote should be valid, try adjusting the VR2 frequency adjustment trimpot to check if the code becomes valid. You will need to select the learning mode (S1) each time to test this. Use small changes over the full range of VR2 before rejecting the remote as unsuitable.

If the code is accepted as valid, the channel LED will light when the programmed button on the handheld remote is pressed again. For toggle mode, the channel will be on with one press of the handheld button and be off on the next press. For momentary operation, the channel will be on for the timer's duration.

In momentary mode, if the handheld remote button is continuously pressed, the channel will remain on until after the button is released, plus the timer period.

If you find that the unit doesn't operate reliably or only works with certain orientations of the remote, it may be due to reception frequency tolerances. In that case, it's just a matter of altering IC1's frequency with VR2 to improve the IR code detection.

Panel labels

Assuming it's all working correctly, all that remains now is printing out and fitting the front and rear panel labels. They are shown in Fig.12 but are also available as a PDF download from pemag.au/Shop/11/468

Information on making front panel details is available on the Silicon Chip website at pemag.au/Help/FrontPanels

Once you have made the labels, affix them in position and cut out the holes using a sharp hobby knife. For the front panel, insert the LED bezels from the front and insert the LEDs from the rear. The PCB is held in place with No.4 self-tapping screws into the four integral mounting posts at the bottom of the case.

Fig.11 (left): as you adjust VR1 to set the timing for a momentary output, the LEDs will show the current setting like this. Rotate VR1 while holding S3 until the LEDs show your desired output on-time, then release S3.

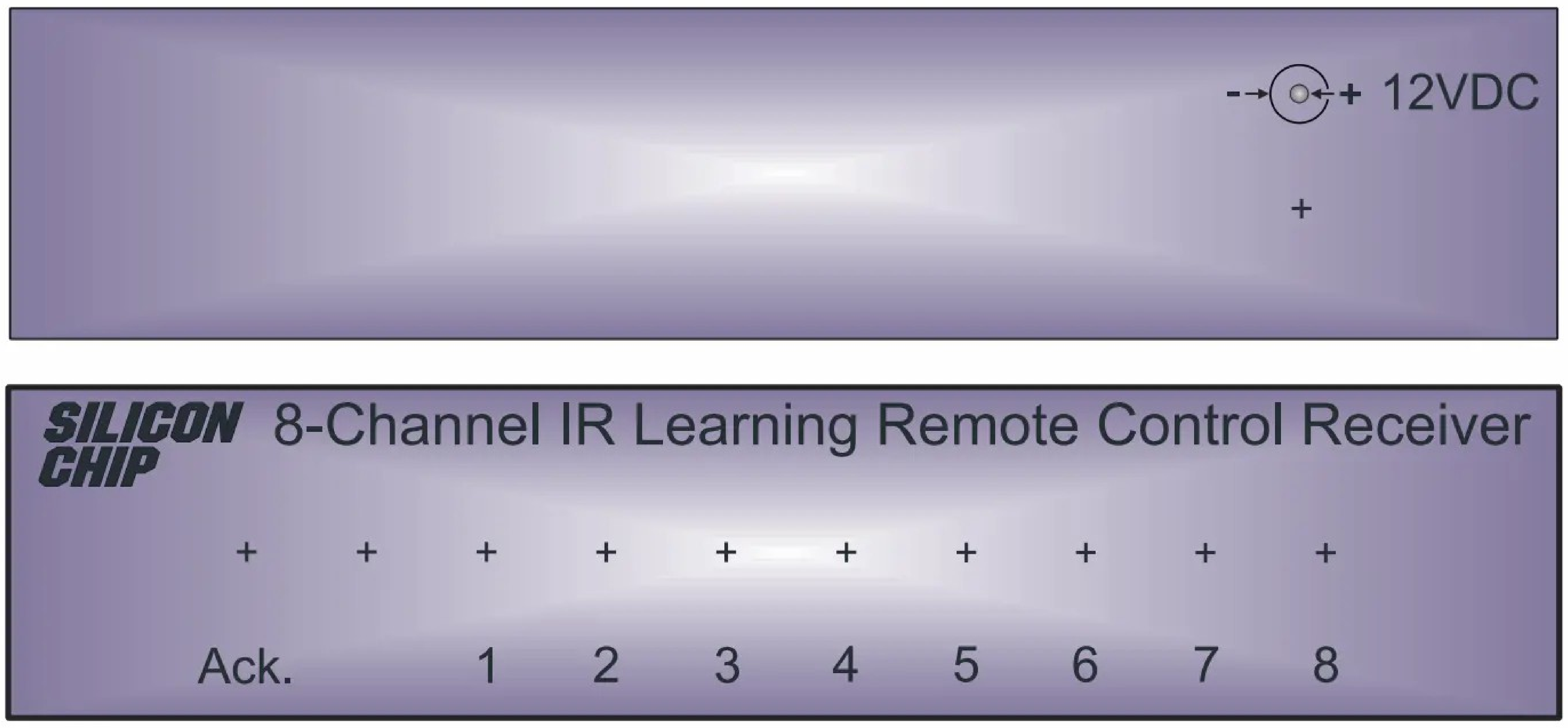
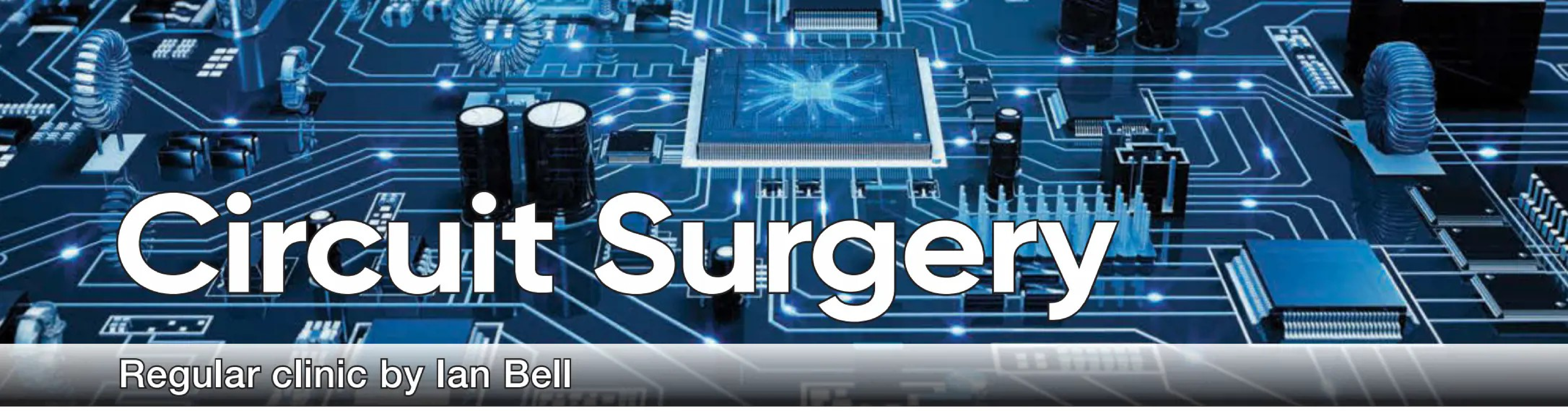


Fig.12 (right): the front and rear panel labels. These can also be downloaded as a PDF from pemag.au/Shop/11/468



Circuit Surgery

Regular clinic by Ian Bell

Topics in digital signal processing – Implementing DSP on a microcontroller, part three

We are looking at various topics related to digital signal processing (DSP).

DSP covers a wide range of electronics applications where signals are manipulated, analysed, generated, stored or displayed as digital data but originate from and/or are converted to real-world signals for interaction with humans or other parts of the physical world.

Fig.1 shows the key elements of a generic DSP system with a signal path from an analog input via digital processing to an analog output.

This does not necessarily represent every DSP system (not all have all the parts shown), but it serves as a reference for the various subsystems we will look at.

In the previous couple of articles, we have been working through an example microcontroller-based DSP implementation. Before getting to the implementation of a sinc filter, we are looking at individual aspects of the system: the digital-to-analog converter (DAC), analog-to-digital converter (ADC), sample timing generation and data transfer.

Last month's article concluded with the DAC controlled by an interrupt timer triggering a callback function at the sampling frequency. This code can easily be adapted to read the ADC in the callback function instead. To illustrate basic operation of the ADC, the values read from the ADC can be fed directly to the DAC (no filtering or other processing will be implemented to begin with).

Both devices use 12-bit samples and operate within the same analog signal voltage range. Therefore, no conversion or adaption is required. If the ADC data is passed directly to the DAC, it should recreate the same signal that was fed to the ADC, subject to the normal limitations of sampled signal processing in terms of maximum input signal frequency and quantisation effects.

We will start with a simple implementation, then consider how the operating speed can be improved using direct memory access (DMA).

ADCs on the microcontroller

The microcontroller on the B-L475E-IOT01A board we are using contains three 12-bit ADCs. These have a maximum sampling rate of 5Msps (megasamples per second). The ADCs have multiple input channels connected to different pins or internal signals on the microcontroller via multiplexers.

The Device Configuration Tool in the STM32CubeIDE app can be used to set which channel each ADC will read from (although the code can change this at any time). ADC1 is most convenient for us because it can be connected to pins PC0 to PC5, which are on Arduino Uno header CN4. This is shown in Figs.2(a) & 2(b), both extracts from the PDF at https://pemag.au/link/ac5j

No other circuitry is connected to these pins.

The default configuration of a new STM32CubeIDE project for the B-L475E-IOT01A board already has pins PC0 to PC5 configured as ADC1 inputs. All that is needed then is to configure ADC1 to use the required pin. We will connect ADC1 channel 1 (IN1) to pin PC0 (CN4 header pin A5). This can be done by setting the Mode for IN1 of ADC1 to "IN1 Single-ended" using the Configuration Tool in STM32CubeIDE, as shown in Fig.3 (the past two articles explain how to do this in more detail).

The DAC's update time of around 3µs indicates a maximum sampling rate of 330kHz. This is slower than the ADC, so the DAC limits the speed of a combined system. This sampling rate is more than adequate for experimenting with signals in the audio range, which is what we will focus on.

Common sampling rates for audio include 44.1kHz (CD), 48kHz (DVD), 96kHz and 192kHz. A minimum sampling rate of around 40kHz is the Nyquist rate for the highest frequency human hearing can perceive, about 20kHz. Higher sampling frequencies reduce the demands on anti-aliasing and reconstruction filters (as discussed in earlier articles) and may increase fidelity.

To observe ADC operation, a signal must be applied to it. Any signal generator capable of delivering signals in the appropriate voltage and frequency range could be used. This includes standalone lab signal generators, those built into oscilloscope and even computer sound cards.

Ideally, we want to automate measuring the frequency response by coordinating signal generation with amplitude measurement. We will discuss this later.

ADC input protection

The microcontroller's ADC can handle voltages from 0V to 3.3V (the supply voltage of the microcontroller and on-chip ADCs). This should be well within the capabilities of a lab signal generator, but the signal from other sources may need amplification or attenuation.

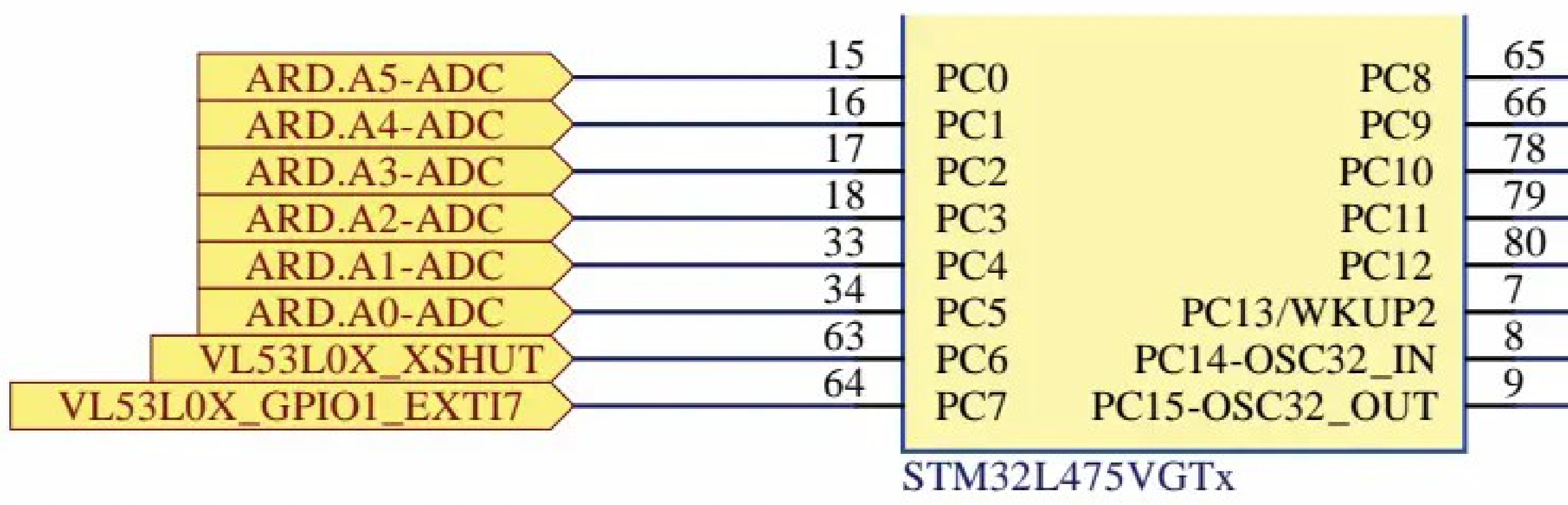


Fig.2(a): the ADC-capable pins on the STM32 microcontroller.

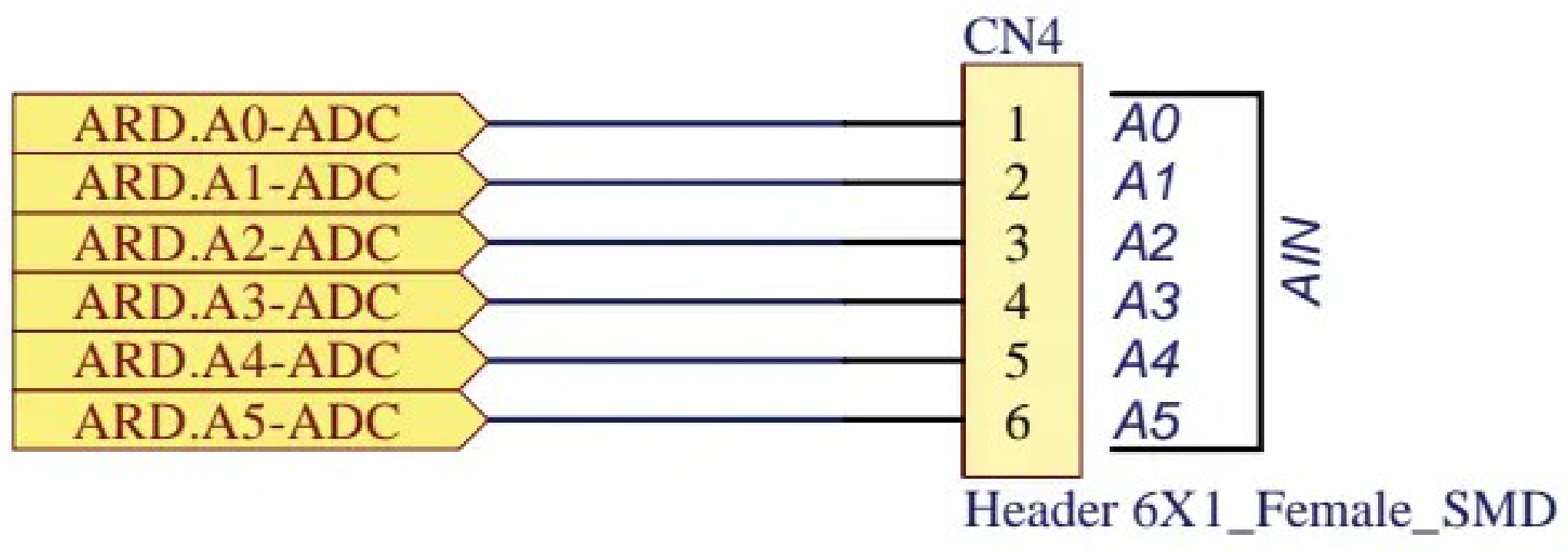


Fig.2(b): the ADC-capable pins are routed to header CN4.

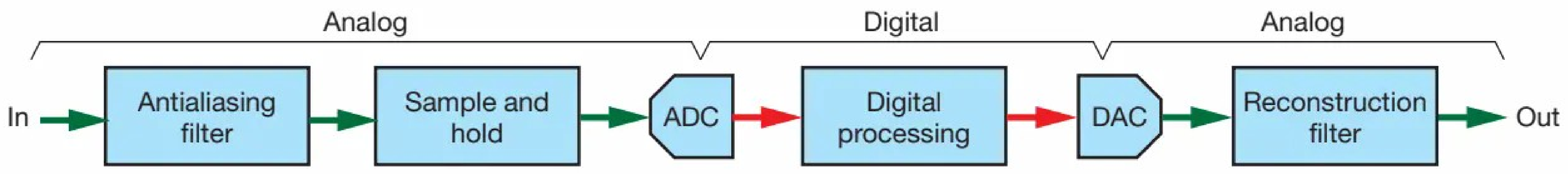
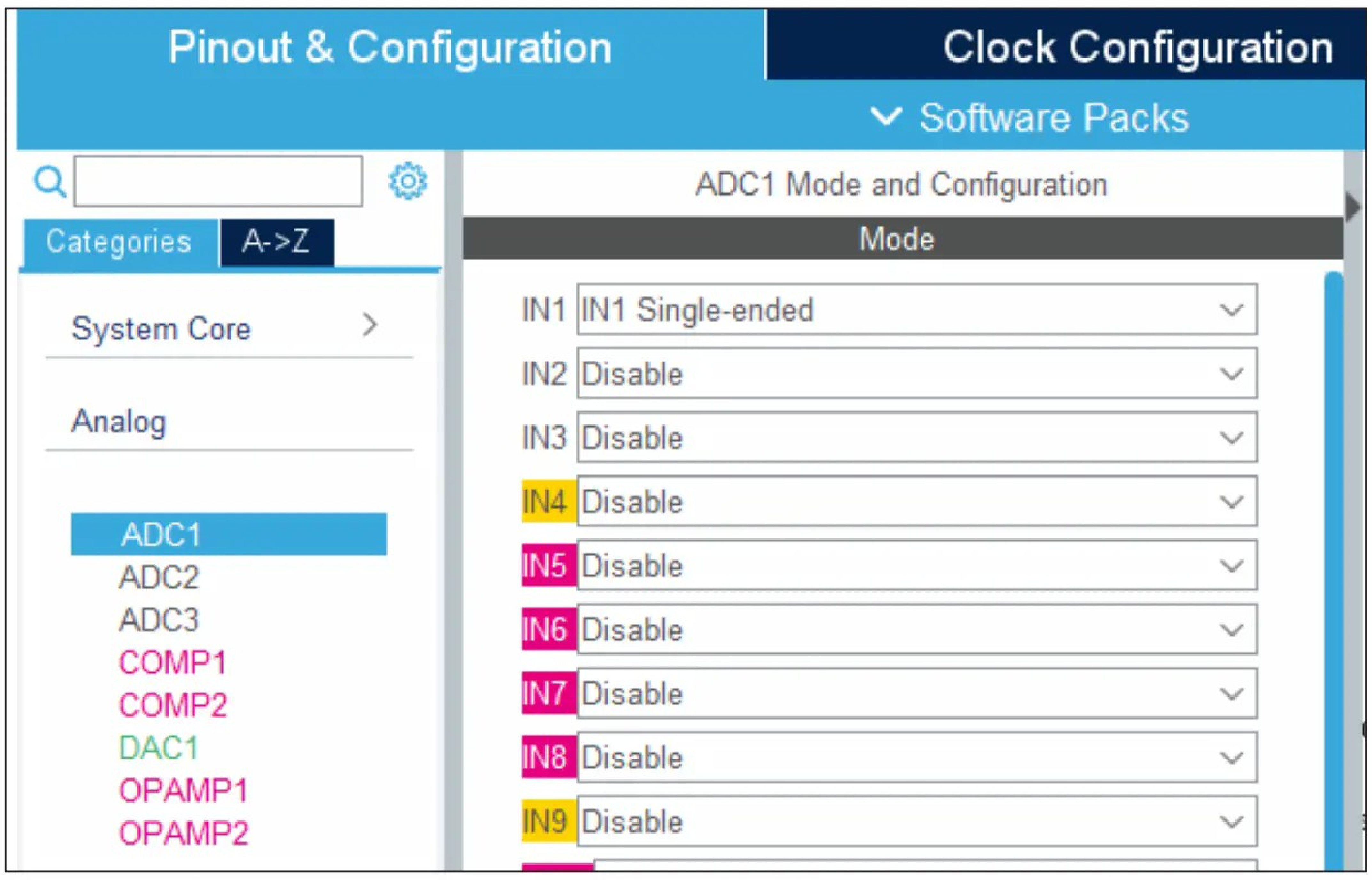


Fig.1: a generic digital signal processing (DSP) system structure.



IN1 IN1 Single-ended
IN2 Disable
IN3 Disable
IN4 Disable
IN5 Disable
IN6 Disable
IN7 Disable
IN8 Disable
IN9 Disable

Fig.3: enabling the ADC1, IN1 input in the STM32CubeIDE Configuration Tool.

Accidentally applying a voltage outside the 0-3.3V range is a possibility when experimenting and this may damage the chip. One solution to this problem is to use diodes to clamp the input voltage close to the supply or ground if a potentially damaging input is applied.

Fig.4 shows an LTspice schematic for circuitry that can be used to connect the signal source to the ADC, including a model of the ADC input. Diodes D1 and D2 provide the input voltage protection; other parts of the circuit will be discussed shortly.

If the input voltage (signal InADC) goes above 3.3V or below 0V by more than the diode forward (switch-on) voltage, the relevant diode conducts preventing significant further voltage excursion at the ADC input. Schottky diodes (such as BAT54) are suited to this purpose due to their low forward voltage (about 0.3V).

Ordinary silicon diodes can be used, but have a have a larger forward voltage (about 0.7V). The absolute maximum input voltage for most inputs on the microcontroller is 4V, so either type of diode would work in this situation, but silicon diodes would be marginal.

Schottky diodes have higher leakage currents, which can cause problems with linearity when protecting signal inputs, but the lower forward voltage provides better protection.

A resistor in series with the signal generator output (R3 in Fig.4) limits the current in the diodes to a safe level if a high voltage is applied. The required value depends on the level of overvoltage that might occur and the diode's current handling. However, there may also be an impact on ADC speed and accuracy, which we will return to.

A possible problem with the diode clamp is that the supply has to sink current during an overvoltage input. If it is not able to sink sufficient current, the supply voltage can be pulled up by the input signal. If this is expected to be a problem, a zener diode (D3 in figure 4) can be used to keep the voltage on the supply from rising too much.

In the schematic, D3 is set up as a 3.9V zener using the quick "ako" (a kind of) method of borrowing a different model (1N750, 4.7V) and changing the breakdown voltage parameter (there is no 3.9V zener in the library).

This does not necessarily give a fully realistic model; however, the diode never conducts in the simulation shown as it is connected across an ideal 3.3V source (the microcontroller/ADC supply). It is included to illustrate the option of using a zener diode here.

ADC input signal

Typically, the input to a DSP system is an AC signal centred on 0V, so the signal has both positive and negative peaks. The ADC on the development board we are using has an input range of 0-3.3V, so a DC offset needs to be added to the AC signal – the result should be a signal centred on half the ADC input range (3.3V ÷ 2 = 1.65V).

There are various ways to achieve this; for example, lab function generators can usually apply a DC offset. However, this is not possible with an audio line output, such as from a PC sound card.

An alternative is to remove any DC from the input with a coupling capacitor (C1 in Fig.4) and set the ADC input DC level to half the supply voltage using an equal-resistor potential divider (R1 and R2). The signal's peak voltage must be less than 1.65V to avoid clipping (3.3V peak-to-peak).

An op amp circuit could also be used to apply the offset, which would buffer the source signal, but that is more complex and unnecessary for running simple tests.

The potential divider resistors load the source, so their values should not be too low. Their specific values are not particularly critical, and could range from 10s to 100s of kilohms. The potential divider resistors, resistor R3 and the coupling capacitor (value C) form a low-pass filter, in which R1 and R2 act in parallel.

The effective resistance (R_P) is half the potential divider resistor value (50kΩ for Fig.4), ignoring R3, which does not have a significant impact. The cutoff frequency is $1 \div (2\pi\ R_P\ C)$, which is 3.2Hz for the values shown, which is below the low end of the audio range. So audio signals will not be affected to any great extent.

The parallel R1/R2 combination also forms a potential divider for the AC input signal with the current-limiting resistor, R3. If R3 is much smaller than R_P, this effect is minimal.

ADC sampling

Fig.4 includes a representation of the ADC's input (inside the dashed box). This comprises the sampling capacitor (C_{ADC}), switch and the resistance (R_{ADC}) of wiring and switching networks between the microcontroller's pin and sampling capacitor. The STM32L475xx datasheet specifies C_{ADC} as 5pF, but not R_{ADC}; however, other sources indicate 6kΩ.

The switch in the schematic is not a simulation switch model – this is just an open circuit as far as LTspice is concerned, so the simulation reflects the state of the circuit when the ADC is not taking a sample.

When the ADC takes a sample, the switch closes briefly, and the sampling capacitor is charged. For the ADC to obtain an accurate result, the sampling capacitor (C_{ADC}) must have sufficient time to charge to the input voltage. It charges from the signal source via any resistance in the signal path.

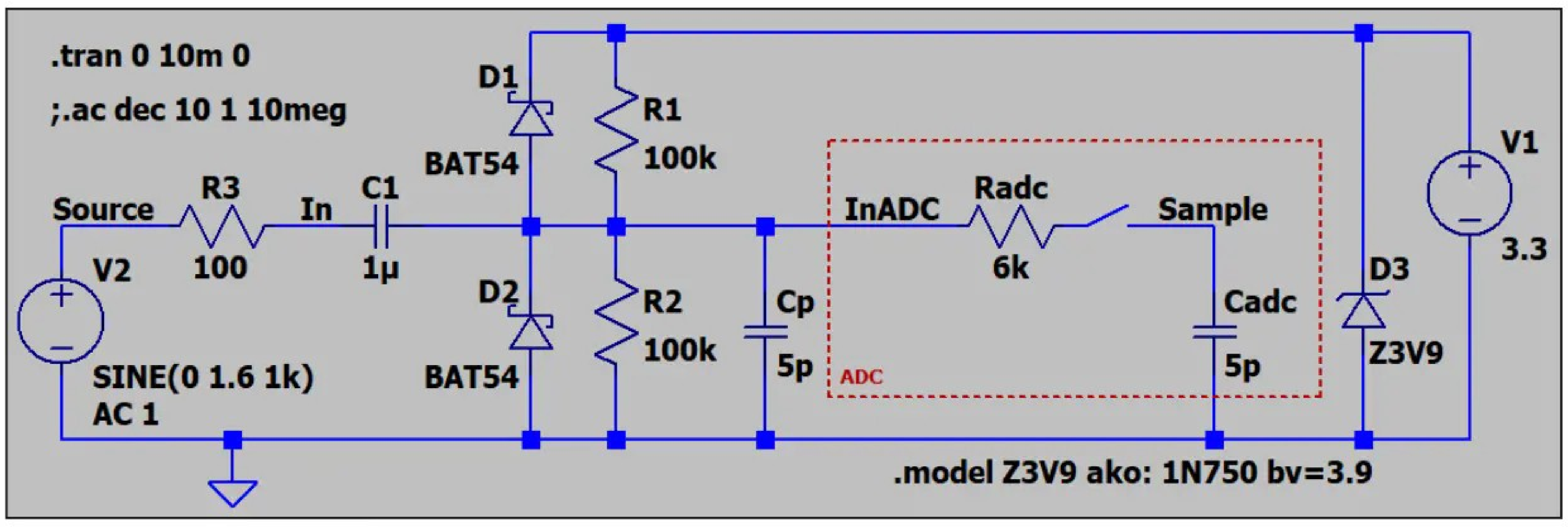
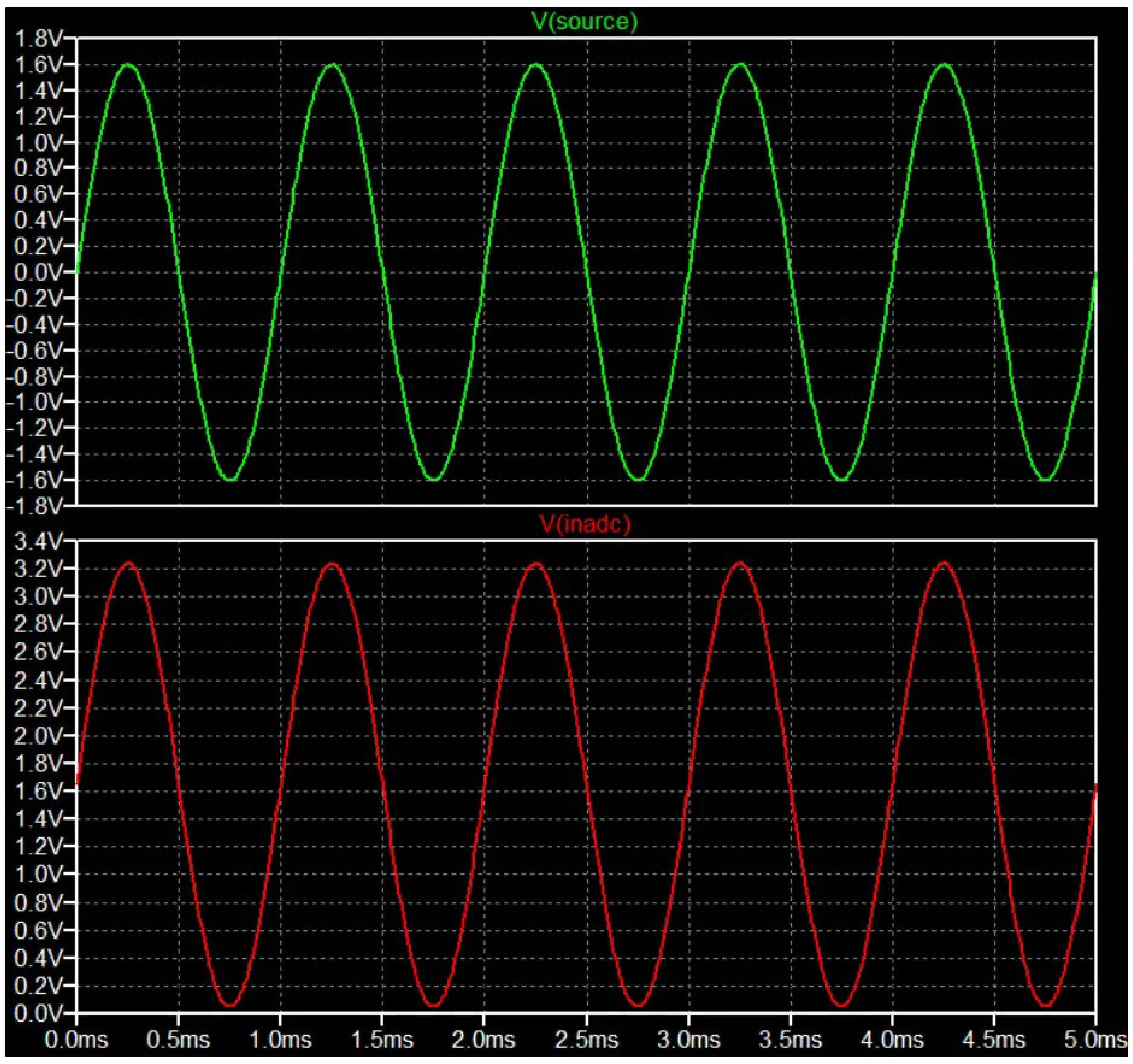
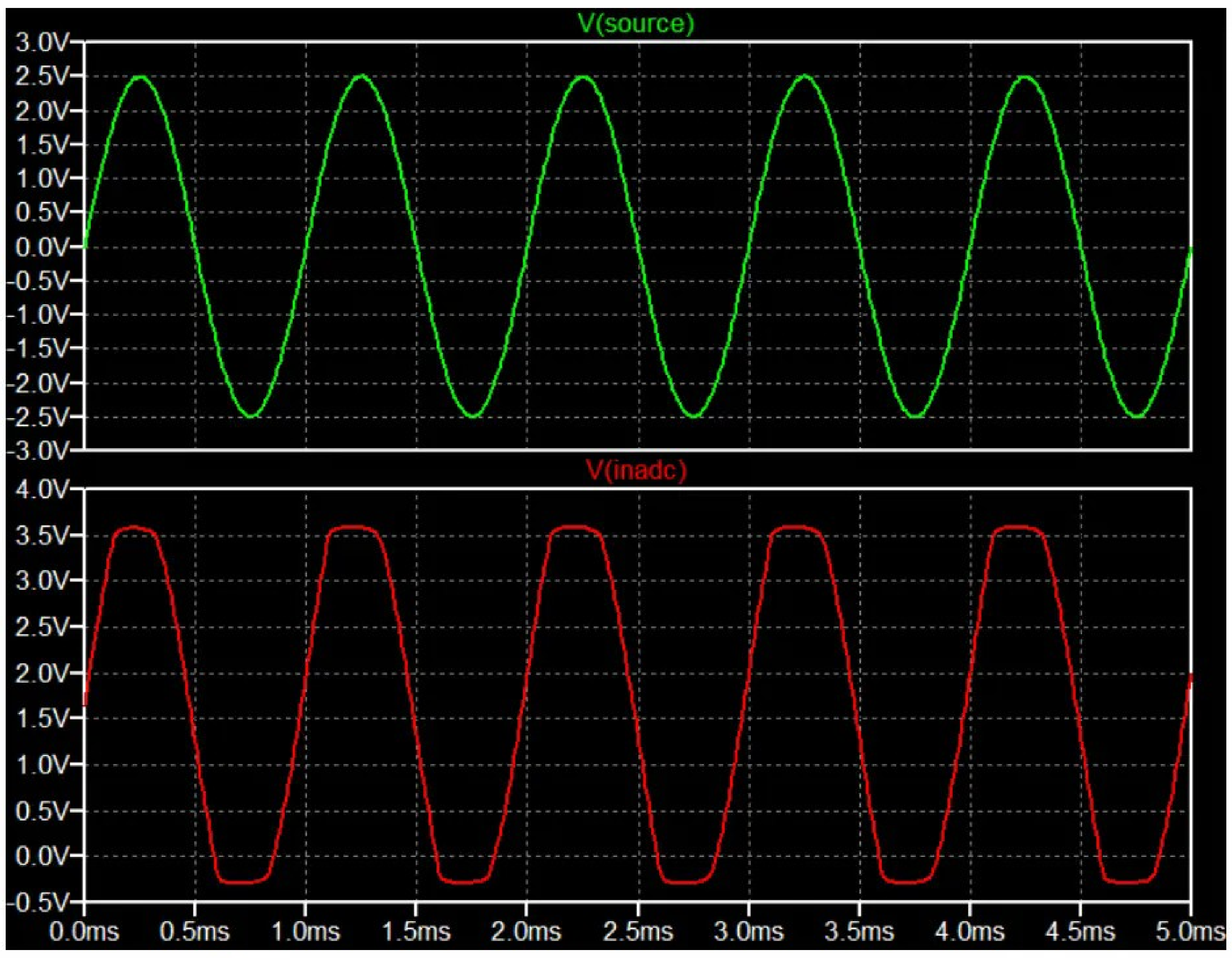

```
.tran 0 10m 0
;.ac dec 10 1 10meg
.model Z3V9 ako: 1N750 bv=3.9
```

Fig.4: an LTspice schematic of ADC input protection, biasing and an ADC input model.



◄ Fig.5: simulation results from circuit in Fig.4, with a close-to-maximum amplitude input signal to the ADC.



▲ Fig.6: simulation results from circuit in Fig.4, with the overvoltage clipped by protection diodes

This includes the source impedance, any inserted resistors (eg, R3 in Fig.4) and the on-chip resistance of the sampling switch, wiring and signal multiplexers (R_{ADC}). There is also a parasitic capacitance of the wiring (C_P in Fig.4), which must be charged to the source voltage before the sample is taken.

The value of C_P in Fig.4 is somewhat arbitrary – in reality, it will depend on the physical structure of the circuit.

The sampling time required depends on the desired accuracy – the more bits in the conversion, the closer the voltage on C_{ADC} must get to the source voltage. Looking at this the other way around, given a specific sampling time and required accuracy, there is a certain maximum resistance that can be tolerated.

The STM32L475xx datasheet (**https://pemag.au/link/ac5i**) specifies the maximum external resistance (designated R_{AIN}) for a range of scenarios – specifically, different sample times and the required resolution (the ADC has 12 bits, but you do not have to use the full resolution).

If the default sampling time is insufficient, the time can be set using the microcontroller's SMPR registers. For the default fastest sampling rate at 12 bits of resolution, the maximum R_{AIN} is 100Ω (as used for R3).

Simulations

Fig.5 shows the results of a transient simulation of the circuit in Fig.4 with a 1kHz, 3.2V peak-to-peak sinewave source signal centred on 0V. The DC shift to 1.65V can be observed on the InADC signal. This is close to the maximum amplitude input.

Fig.6 shows the same signals with a source amplitude of 5V peak-to-peak (V1 sine amplitude at 2.5V). This causes the ADC input to be clipped by the protection diodes, limited at -0.3V and +3.6V.

Fig.7 shows the frequency response of the ADC input network. This confirms the low-frequency -3dB point is at about 3.2Hz. The response starts to decrease just below 10MHz due to the parasitic capacitance, C_P.

As noted, this has a somewhat arbitrary value here, but unless there is a large capacitance in the circuit, it will not limit the maximum frequency which can be processed by the ADC. That is more likely to be determined by the achievable sampling rate.

Using the DAC & ADC together

Last month, we finished with an STM32CubeIDE project in which a code-generated waveform was produced by the DAC. The sample timing was controlled by interrupts from a hardware timer.

This example will use the same approach for timing and DAC use. The timer and DAC configuration are the same as the final project from last month, while the ADC is configured as described above and shown in Fig.3.

The previously used DAC set and start values are not needed in this project, but a variable (`Value`) is required to transfer the sample value from the ADC to the DAC (Listing 1, line 62). This is initialised to half the 12-bit range (corresponds with zero signal input).

Variables are also declared for the timer prescaler, period and frequency (Listing 1, lines 64-66). These values are used to reinitialise the timer to obtain the required sampling period (Listing 1, lines 137-144) in the Use Code 2 section, as was discussed last month. The DAC and timer are also started in this section (Listing 1, lines 133 and 146), again as described last month.

The DAC is updated by code that runs when a timer interrupt occurs. To achieve this, the code is placed in an interrupt service routine (ISR) function named **`HAL_TIM_PeriodElapsedCallback()`**, which is added to the User Code 4 section near the end of the autogenerated **`main.c`** file (Listing 2).

The code in the function must check which timer caused the interrupt before dealing with it. This is done by checking the identity of the timer handle passed to the function (Listing 2, line 833), which in this case is Timer 6.

After confirming Timer 6, the code in the ISR reads the ADC and passes this value directly to the DAC using the **`Value`** variable (Listing 2, lines 836 to 843). As

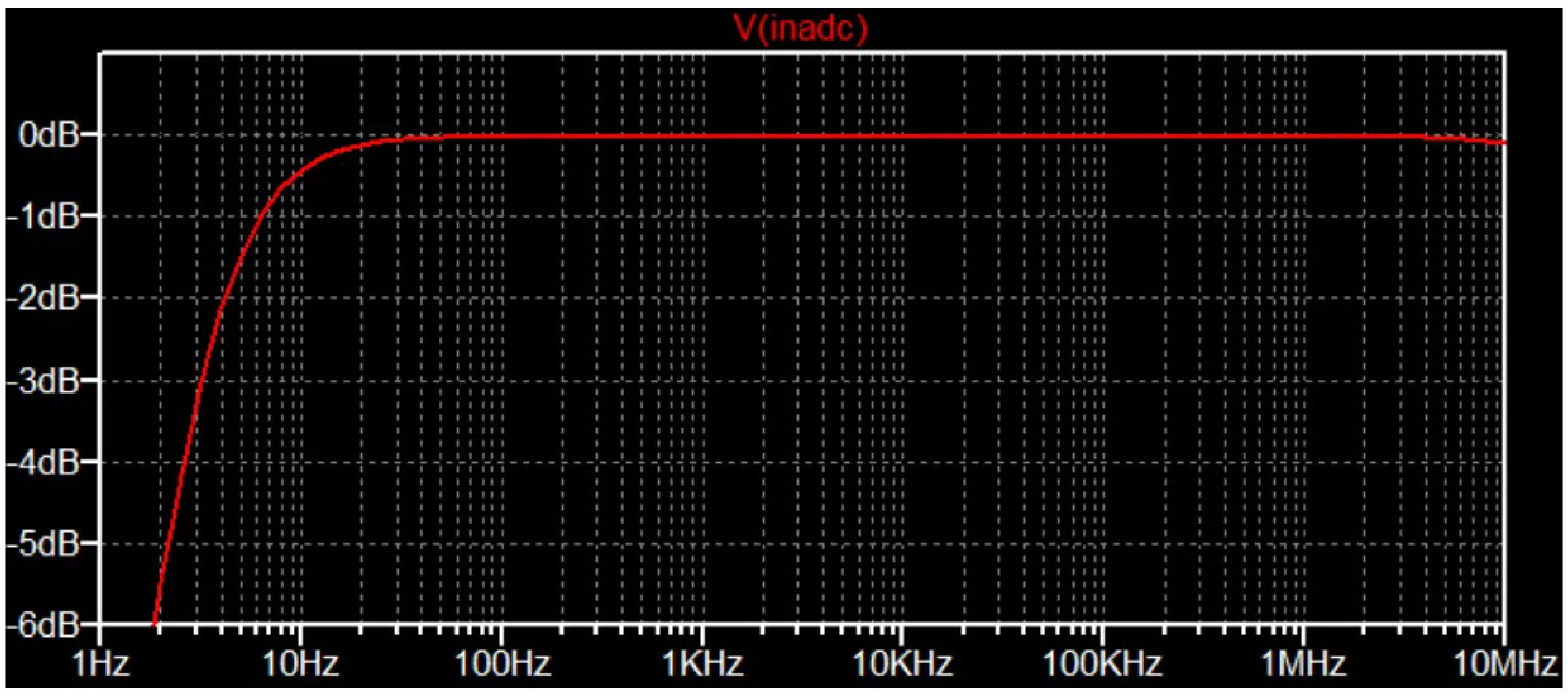


Fig.7: the simulated frequency response of the ADC input circuit in Fig.4.

in previous examples, a GPIO pin is toggled when the DAC is updated to allow us to observe the timing of the sampling process.

To get an ADC reading directly under program control, it is necessary to call the **HAL_ADC_Start** function each time a conversion is required (at the time the input sample it to be taken). The function only requires one parameter: the ADC handle (**hadc1** in this case).

The conversion process is not instantaneous; the result can only be obtained after the conversion has finished. Calling the function **HAL_ADC_PollForConversion** will wait for the ADC to complete the current conversion. The function has two parameters: the handle and a timeout value in millisecond.

After the timeout period, the function will stop waiting for the ADC and the code will continue running even if the ADC has not responded. Using the #defined value **HAL_MAX_DELAY** for the timeout parameter effectively sets an infinite timeout period.

Once it returns from the function call to **HAL_ADC_PollForConversion**, the **HAL_ADC_GetValue** function can be used to obtain the conversion result. This function only takes the ADC handle as a parameter.

The three functions mentioned above are called in sequence in the timer interrupt callback function (ISR). They replace the waveform generation code in the previous projects. The code to write to the DAC and generate a pulse on the GPIO output is the same as in DAC projects.

Program execution results

The DAC output from running the code in Listings 1 & 2 was obtained using an oscilloscope and is shown in Fig.8. The sampling frequency was 48kHz, as per Listing 1. A 3.2V peak-to-peak 2.5kHz sinewave, with 0V DC offset, was applied to the ADC input via the circuit shown in Fig.4 (using R1, R2, R3, C1, D1 & D2).

Fig.8 shows the signal source and the DAC output. There is no filter on the DAC output, so we can see the stepped waveform. We can see that the input varies around 0V, and the DAC output has a 1.65V offset. Fig.9 shows the DAC output and timing pulse, for which the oscilloscope measured a frequency of 47.99kHz – within the expected crystal precision.

```
/* USER CODE BEGIN PV */
uint32_t Value = 2048;  // Current signal value
uint32_t Prescaler = 1; // Timer prescaler setting
float Period;           // Timer period in us
float Frequency = 48.0; // Sample frequency in kHz
/* USER CODE END PV */
…
 /* USER CODE BEGIN 2 */
 HAL_DAC_Start(&hdac1, DAC_CHANNEL_2);

 // Calculate TIM6 counter period (ARR) from required DAC sample frequency
 // Clock frequency is 80 MHz
 Period = (80000.0/Prescaler)/Frequency; // F in KHz
 htim6.Init.Prescaler = Prescaler-1;
 htim6.Init.Period = Period-1;
 // Reinitialise TIM6 with new values
 if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
 {
   Error_Handler();
 }

 HAL_TIM_Base_Start_IT(&htim6);

 /* USER CODE END 2 */
```

Listing 1: variables and user initialisation code for the ADC to DAC transfer.

```
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim == &htim6)
  {
    // Read the ADC to get the value to output to the DAC
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,HAL_MAX_DELAY);
    Value = HAL_ADC_GetValue(&hadc1);

    // Update the DAC and pulse the D8 pin to help measure timing
    HAL_GPIO_WritePin(GPIOB, ARD_D8_Pin, GPIO_PIN_SET);
    HAL_DAC_SetValue (&hdac1, DAC_CHANNEL_2, DAC_ALIGN_12B_R, Value);
    HAL_GPIO_WritePin(GPIOB, ARD_D8_Pin, GPIO_PIN_RESET);
  }
}
/* USER CODE END 4 */
```

Listing 2: reading values from the ADC and writing them to the DAC.

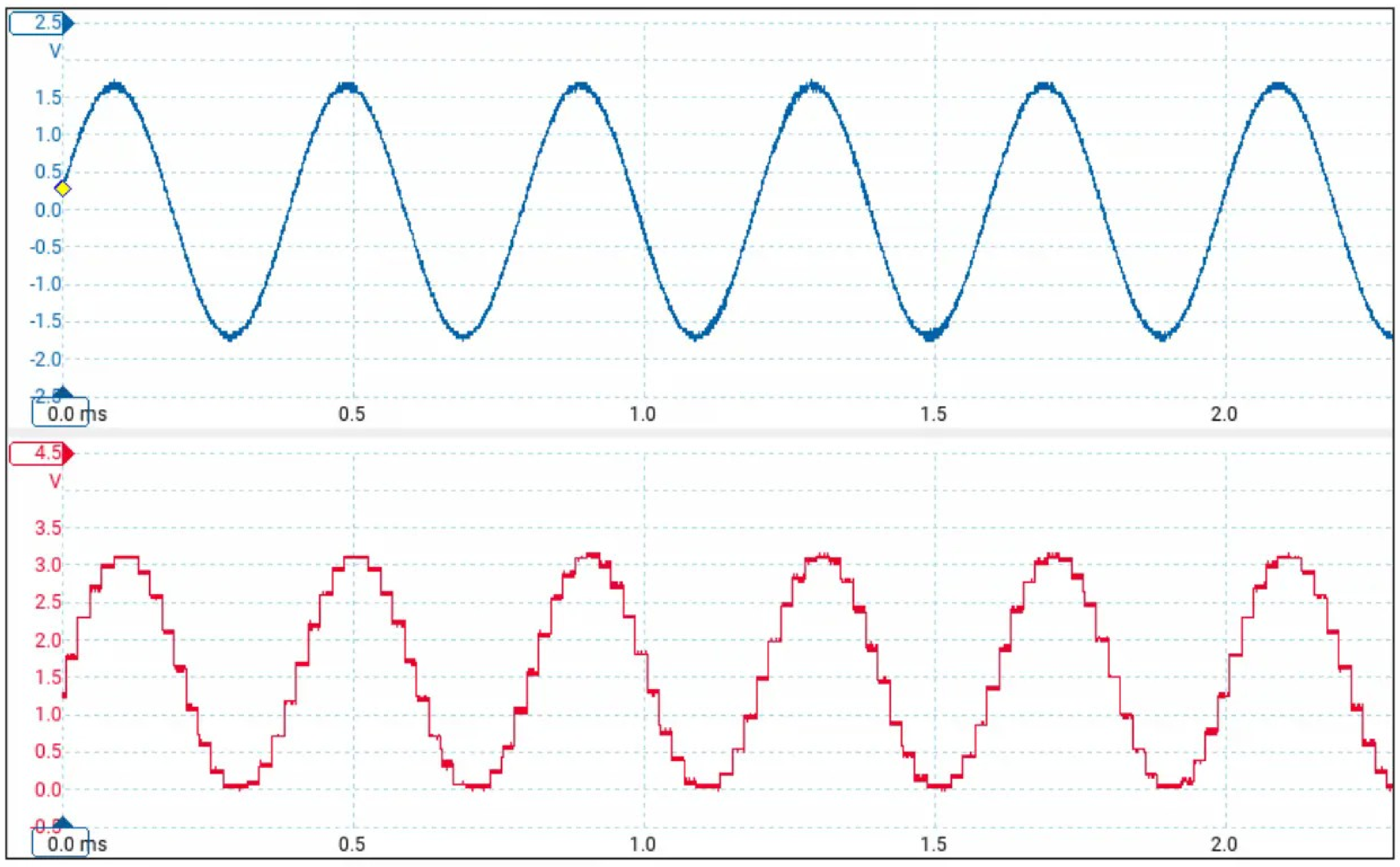


Fig.8: the ADC input (blue) and DAC output (red) signals using the code in Listings 1 & 2.

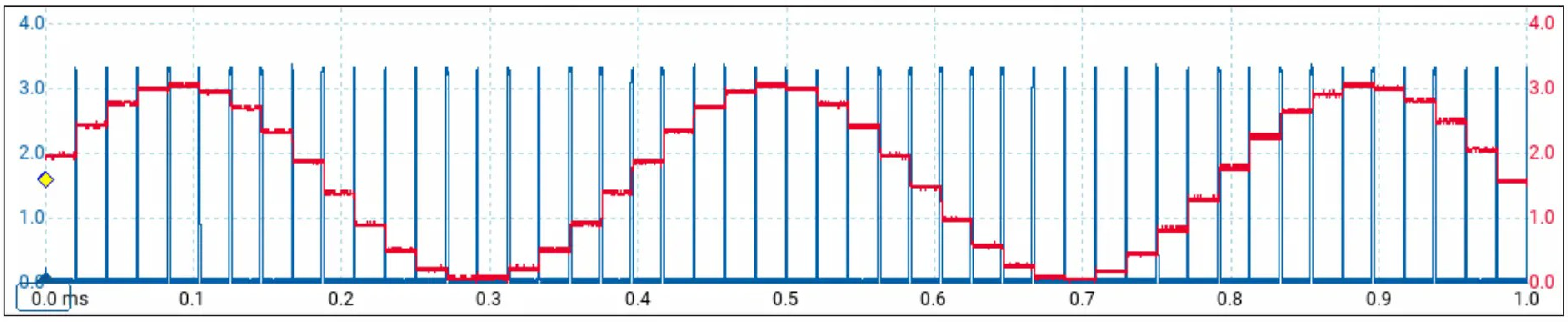


Fig.9: how the timing pulses (blue) relate to the DAC output (red).

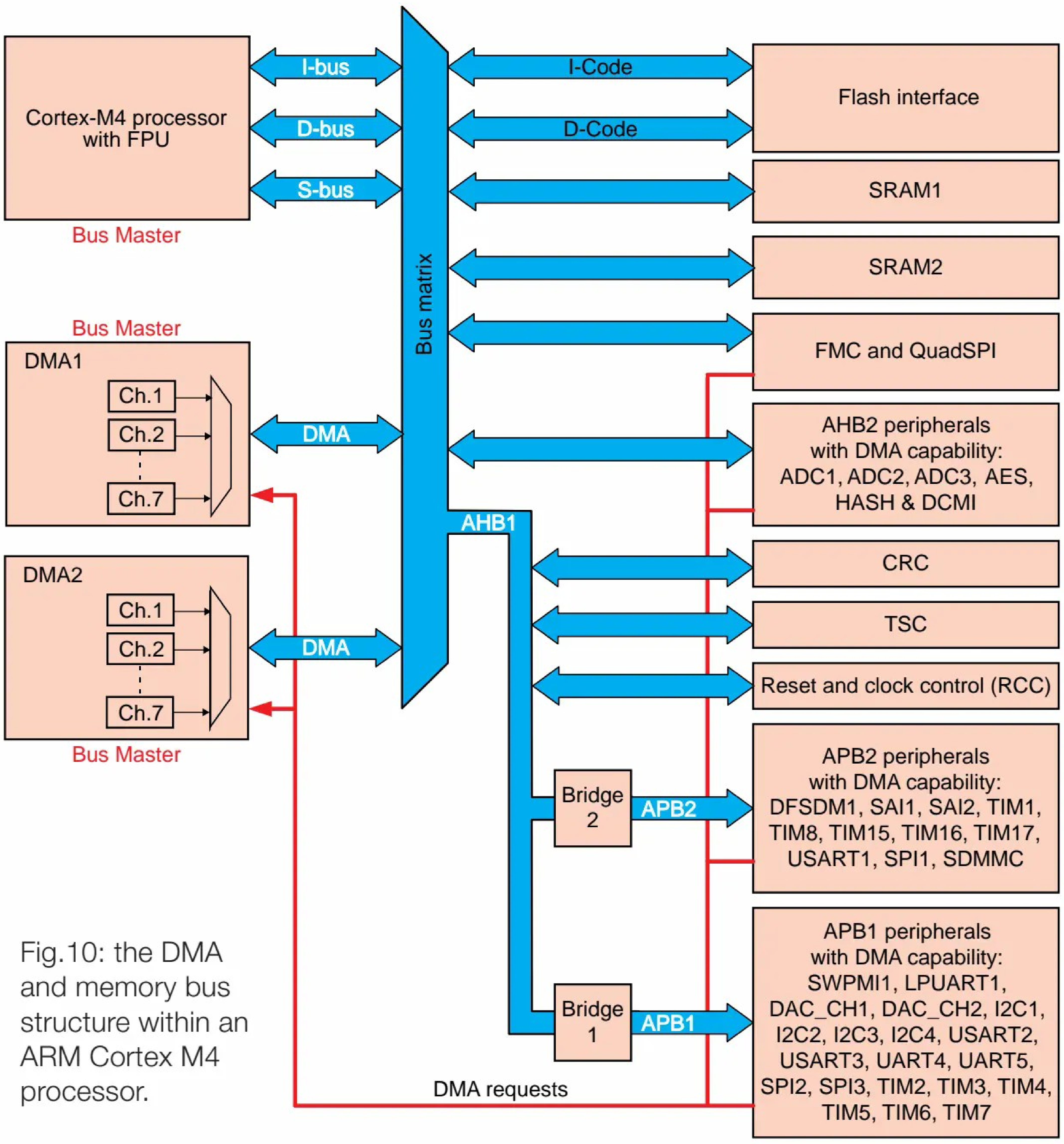

Cortex-M4 processor with FPU
Bus Master
I-bus
D-bus
S-bus
Bus matrix
I-Code
D-Code
Flash interface
SRAM1
SRAM2
FMC and QuadSPI
AHB2 peripherals with DMA capability: ADC1, ADC2, ADC3, AES, HASH & DCMI
Bus Master
DMA1
Ch.1
Ch.2
Ch.7
DMA
AHB1
CRC
TSC
Reset and clock control (RCC)
DMA2
Ch.1
Ch.2
Ch.7
DMA
Bus Master
Bridge 2
APB2
APB2 peripherals with DMA capability: DFSDM1, SAI1, SAI2, TIM1, TIM8, TIM15, TIM16, TIM17, USART1, SPI1, SDMMC
Bridge 1
APB1
APB1 peripherals with DMA capability: SWPMI1, LPUART1, DAC_CH1, DAC_CH2, I2C1, I2C2, I2C3, I2C4, USART2, USART3, UART4, UART5, SPI2, SPI3, TIM2, TIM3, TIM4, TIM5, TIM6, TIM7
DMA requests

Fig.10: the DMA and memory bus structure within an ARM Cortex M4 processor.

Experiments with increasing the sampling frequency showed a limitation of this example. The maximum sampling frequency is just over 60kHz, although we know the converters can operate faster than this.

The speed is limited by the individual software-controlled ADC conversions – the time taken for the processor to interact with the ADC and DAC (initiating ADC conversion, polling for conversion and reading ADC, writing DAC).

Going faster with DMA

In the previous example, the code initiates ADC conversion, waits for it to complete, reads the ADC value into memory and then writes it to the DAC. All of this takes significant time, which could otherwise be spent doing things that make better use of the processor's capabilities – such as applying signal processing functions to the data.

Using DMA, it is possible to eliminate the need for the processor to perform ADC-to-memory and memory-to-DAC data transfers. DMA operates in parallel with the processor's activities, using separate hardware. The DMA hardware controls the read/write processes for the data converters and memory, leaving the processor free to do other things.

With DMA, the processor has more time to perform signal processing operations, and the sampling rates can be faster.

In general, the movement of data to and from memory with DMA can take place as a one-off process on a single batch of data (normal mode) or as a continuous process (circular mode). Continuous circular operation is generally required for real-time DSP applications.

In this mode, the DMA system constantly reads/writes from/to a block of memory (often referred to as a buffer), with each sample read from, or written to, the next address in sequence. When the write or read address reaches the end of the buffer, it returns to the start – it constantly circles through the set of memory locations in the buffer, hence 'circular mode'.

Processor bus architecture

Processors use buses to interact with their memory and peripheral devices, such as ADCs and DACs. A bus is a collection of signals arranged so that multiple subsystems can communicate with each other.

Early and very basic processors may use the well-known shared bus based on tristate buffers (buffers able to drive a line high, low or not drive it at all). The processor sets up an address on the bus's address wires and asserts a read or write signal. The Data to or from the processor is placed on the bus's data wires via tristate buffers.

Tristate outputs allow multiple devices to put data on the same wires – the design has to ensure only one data source is outputting to the data lines at any time.

Tristate buses are suited for basic systems where the processor, memory and peripherals are separate chips or boards, as they are efficient in terms of wiring and connectors. The buses on modern system-on-chip (SoC) devices (such as advanced microcontrollers) are significantly more complex and use a different approaches.

Fig.10 shows the microcontroller's bus and DMA architecture, an annotated version of the DMA block diagram in ST's RM0351 Reference manual (**https://pemag.au/link/ac6r**), which covers various SMT32L4 processors.

In a basic system, the bus is controlled solely by the processor. With more complex systems, more than one subsystem can control the bus – these are known as bus masters. In Fig.10, we see three blocks labelled as bus masters: the processor and two DMA controllers. The ARM M4 processor contains three bus masters, for instruction (I), data (D) and system (S) (for peripheral) operations.

The Bus Matrix is a large routing switch that allows connections to be made from the bus masters to the other systems (memory and peripherals). This is a more effective and lower-power approach than a tristate bus – the complexity of "wiring" involved is much less of an issue on-chip than it would be trying to do a similar thing at a board or backplane level.

When there are multiple bus masters, there is always a possibility that more than one master will want to access a given resource at the same time – the Bus Matrix subsystem controls which master has access at any time, using a process called bus arbitration.

DMA operations are managed by DMA controllers. These are bus masters that operate independently of the processor (although what they do is configured by the processor). They can move data via the bus system separately from the processor so, for example, they can transfer values from the ADC directly into memory, or data from memory directly from the DAC.

DMA transfers can be triggered by events in the peripherals, such a ADC conversion completion. For ADCs and DACs, the timing of the conversions can also be controlled by timers, as we previously did. However, with DMA, this does not require timer interrupts to the processor.

Instead, the timers directly control the sampling (eg, trigger an ADC conversion in hardware), and hence when the DMA transfers occur. The DMA controllers can trigger interrupts; for example, when they reach the end of a block of memory they are accessing, so the processor can handle it or refill the buffer with more data.

Fig.10 shows that there are different

types of bus connecting the peripherals: the AHB (Advanced High-performance Bus) and APB (Advanced Peripheral Bus). AHB is for high-speed transfer, while APB is for lower-bandwidth, lower-power peripherals. A bridge is a subsystem for connecting different buses (buses with different speeds, protocols, number of bits etc).

Using DMA

To use DMA with the ADC and DAC in an STM32CubeIDE project, we need to apply the appropriate configuration settings for the ADC and DAC. The DMA unit has to be enabled, and settings need to be made to configure DMA operation as required for the project. HAL library functions are used to start the DMA process.

Code to handle data coming from the ADC via DMA, or provide values for the DAC via DMA, is placed in HAL library callback functions (similar to those used for timer interrupts).

On the microcontroller chip used here, DMA transfers occur in either word (32-bit), or half-word (16-bit) increments. The code here uses 32-bit integers (uint32_t data type), to be compatible with the HAL ADC and DAC functions. Therefore, the data width used by the DMA needs to be configured to use words to match this.

The ADC and DAC have resolutions of 12 bits, but this data is processed in code using 32-bit values. These values would fit in half-words, but that would require additional, unnecessary processing steps.

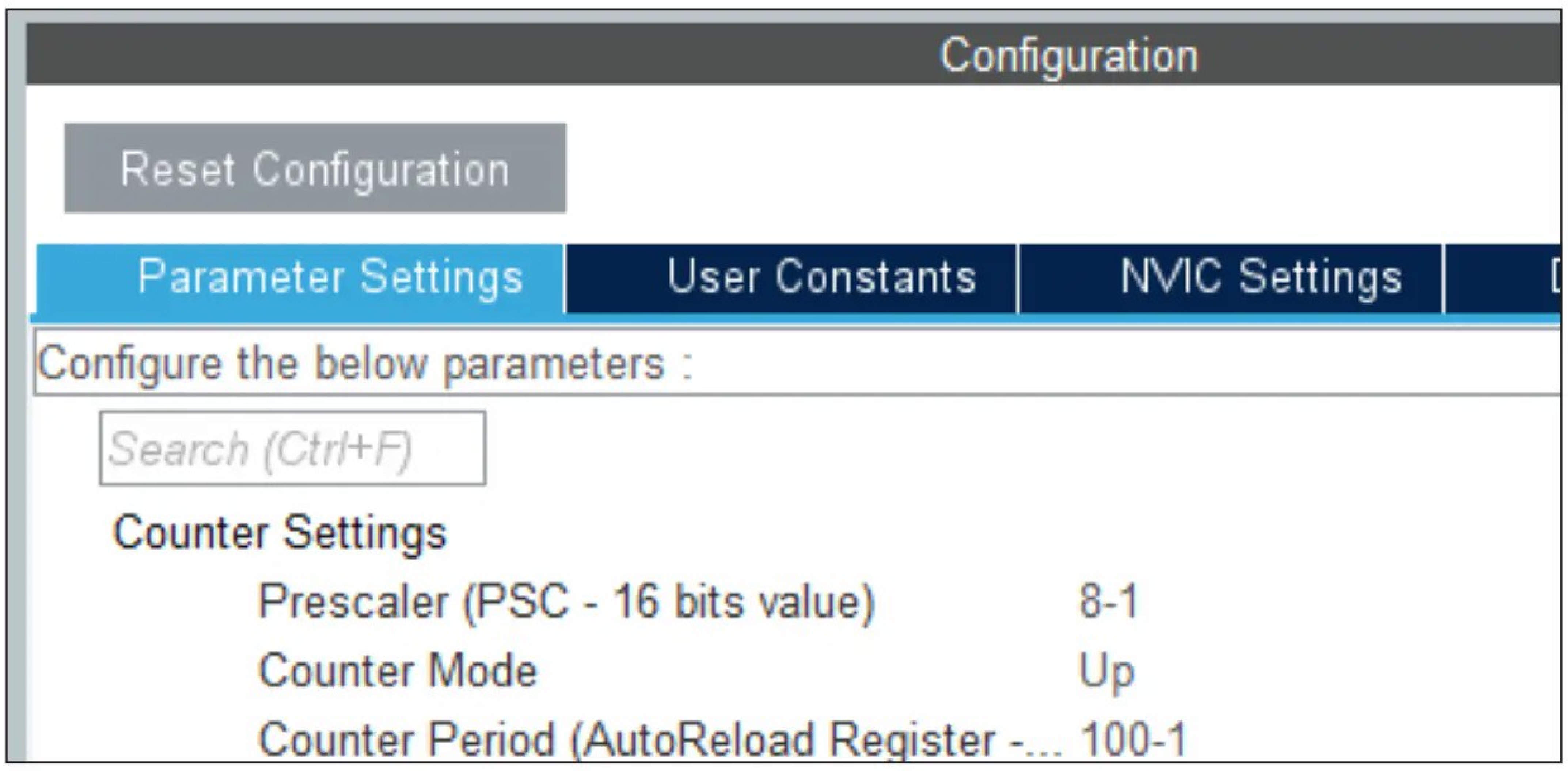
Continuous circular DMA operation is set up via the ADC, DAC and Timer settings in the Configuration Tool. The buffer memory size and location are set up in code. We will use a new project and configure it for DMA.

Timer, ADC & DAC settings

Select Timer 6 (TIM6) in the Configuration Tool and check "Activated" in the Mode section of the Parameter settings, as before. Different timer settings are required in this project to directly trigger the ADC and DAC conversions from the timer. In the Parameter Settings tab in the configuration section, set "Trigger Event Selection" to "Update Event" (see Fig.11).

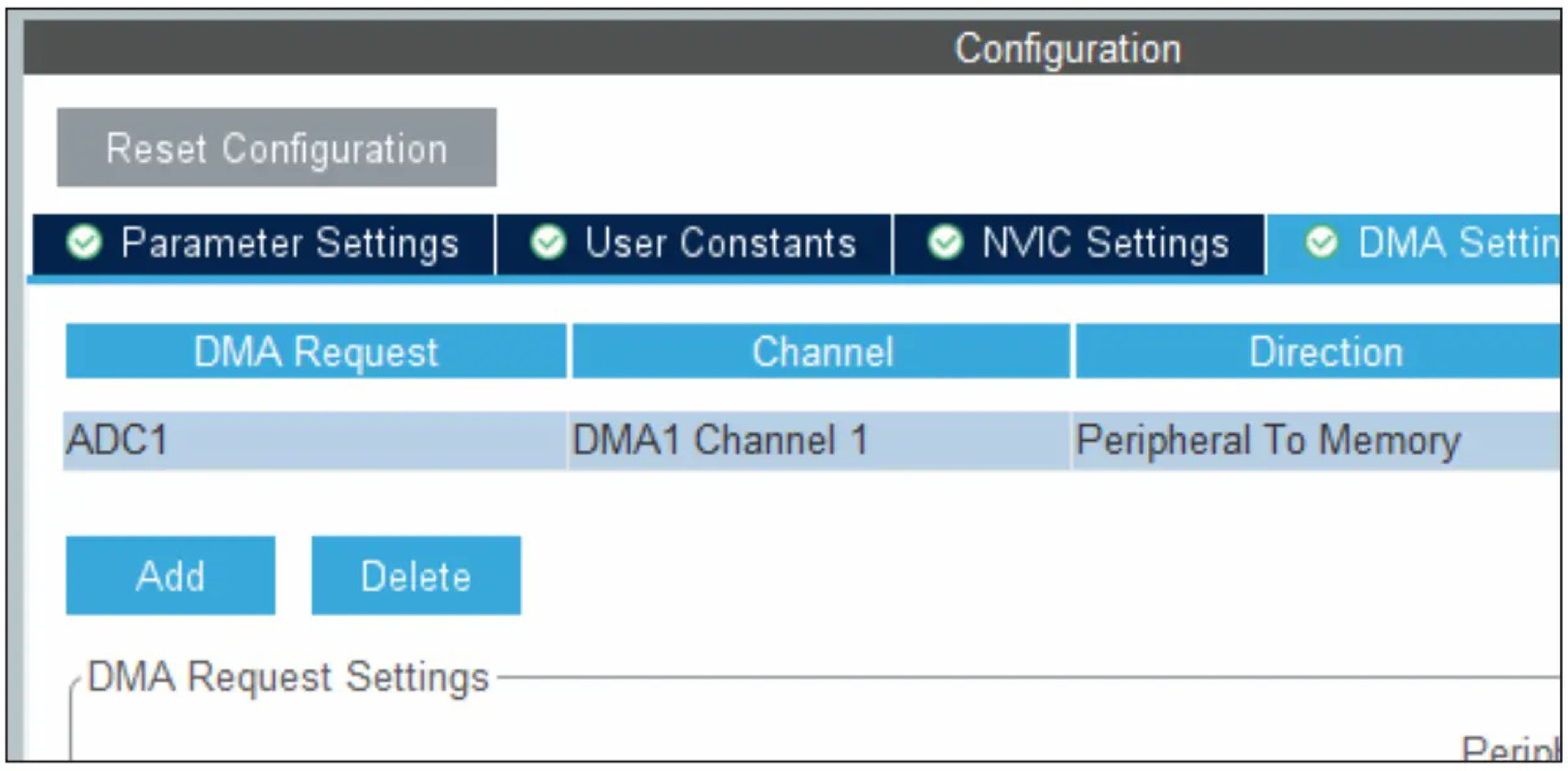
This will enable the timer's trigger output signal, which will be used to activate the ADC and DAC conversions, and hence the DMA transfers of their data. The timer interrupt is not enabled in this example.

The ADC is enabled in Single-ended mode, as in Fig.3. The DMA is configured by clicking the Add button in the DMA settings tab in the ADC configuration (see Fig.12). This will display a drop-down menu under the DMA Request column. Select ADC1 from this.



Counter Settings
Prescaler (PSC - 16 bits value) 8-1
Counter Mode Up
Counter Period (AutoReload Register -... 100-1

Fig.11: the timer settings for our DMA project.



DMA Request	Channel	Direction
ADC1	DMA1 Channel 1	Peripheral To Memory

Fig.12: configuring the ADC1 DMA settings.

In the DMA Request Settings, set Mode to Circular and both "Data Width" values to Word, as discussed before. Also click on Low in the Priority column for ADC1 and change it to High in the drop-down menu.

Click on the Parameter Settings tab in the ADC configuration. Enable "DMA Continuous Requests", then Set "External Trigger Conversion Source" to "Timer 6 Trigger Out event".

The DAC is enabled by setting "OUT2 connected to" to "only to external pin" mode, as previously. In the parameter settings tab of the DAC Configuration, Trigger is set to "Timer 6 Trigger Out event" so the timer triggers the DAC.

The DAC DMA settings are very similar to the ADC – select DAC-CH2 under DMA Request after adding the request and set circular mode, then the word data widths and high priority, as with the ADC.

DMA code

The ADC and DAC both require an area of memory to store the data that is being read (ADC buffer) and written (DAC buffer). In coding terms, these need to be declared as arrays of 32-bit integers in the User PV section (Listing 3, lines 66-67). An ADC buffer and a DAC buffer of the same size are required.

We need to use the buffer size and half size values throughout the code, so these values are #defined in the User PD section to make them easy to change (Listing 3, lines 34-35). This example uses 100-word buffers. The prescaler, period and frequency values are defined and used as in the previous examples (Listing 3, lines 68-70).

```
32  /* Private define ------------------------------------*/
33  /* USER CODE BEGIN PD */
34  #define BUF_SIZE 100
35  #define HALF_BUF_SIZE 50
36  /* USER CODE END PD */
…
65  /* USER CODE BEGIN PV */
66  uint32_t ADC_buffer[BUF_SIZE];
67  uint32_t DAC_buffer[BUF_SIZE];
68  uint32_t Prescaler = 1;     // Timer prescaler setting
69  float Period;               // Timer period in us
70  float Frequency = 96.0;     // DAC sample frequency in kHz
71  /* USER CODE END PV */
```

Listing 3: #defines and definitions for the DMA project.

```
139 /* USER CODE BEGIN 2 */
140 // Start ADC and DAC with DMA and timer to trigger ADC and DAC
141 HAL_ADC_Start_DMA(&hadc1, ADC_buffer, BUF_SIZE);
142 HAL_DAC_Start_DMA(&hdac1,DAC_CHANNEL_2,DAC_buffer,BUF_SIZE,DAC_ALIGN_12B_R);
143
144 // Calculate TIM6 counter period (ARR) from required sample frequency
145 // Clock frequency is 80 MHz
146 Period = (80000.0/Prescaler)/Frequency; // F in KHz
147 htim6.Init.Prescaler = Prescaler-1;
148 htim6.Init.Period = Period-1;
149 // Reinitialise TIM6 with new values
150 if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
151 {
152   Error_Handler();
153 }
154 HAL_TIM_Base_Start(&htim6);
155 /* USER CODE END 2 */
```

Listing 4: the initialisation code for the DMA project.

The DMA controller operations to read data from the ADC and write to the DAC need to be started. This is achieved using two functions; **HAL_ADC_Start_DMA** and **HAL_DAC_Start_DMA**, called during initialisation in the User Code 2 section (Listing 4, lines 141-142).

The **HAL_ADC_Start_DMA** function requires three parameters: the ADC handle, the buffer array and the buffer size. The **HAL_DAC_Start_DMA** function requires five parameters: the DAC handle and channel number, the buffer and buffer size and a data alignment specifier.

The data alignment specifier is the same as that discussed in the first DAC example and the same value, **DAC_ALIGN_12B_R**, will be used here.

Also, the same code as before will be used to configure and start the timer (TIM6) after configuring the sampling rate (Listing 4, lines 144-154).

With DMA running, the processor can read the ADC values directly from memory (from the ADC buffer), but in the code, we need to know which values are valid when, because the buffer is being constantly overwritten with new data by the DMA controller.

This is handled by two callback functions, **HAL_ADC_ConvHalfCpltCallback** (the buffer half-full callback) as well as **HAL_ADC_ConvCpltCallback** (the buffer full callback). A separate callback is not required to write to the DAC – we can write to the DAC buffer during the ADC callbacks as long as everything is correctly coordinated.

Like the timer interrupt code, the DMA buffer callback functions are placed in the User Code 4 section (see Listing 5). In both functions in this example, all the data from the relevant half of the ADC buffer is transferred to the DAC buffer (copying the content of the ADC buffer array to the corresponding location in the DAC buffer array).

This will replicate the ADC input waveform on the DAC output. The code to achieve this is a simply a for-loop iterating over all the array locations in the relevant half of the buffer and performing the copy operation (Listing 5, lines 860-863 and 870-873). No code is required to directly read the ADC or write to the DAC as this is done by the DMA hardware.

The DMA callbacks work similarly to the timer interrupt callbacks in previous examples. The function names are predefined, and users write their own code in functions with these names to implement the operations required when the relevant events occur.

As the name implies, the ADC buffer half-full callback runs when the first half of the ADC buffer has been filled with new data. For a while after this time, the data will not change in the first half of the buffer (because the second half is being written to). During this time, data from the first half of the buffer can safely be read by the processor.

The ADC and DAC buffers are running in synchronicity – the buffers are set up to be the same size, and the configuration setup detailed above causes the ADC and DAC conversions to be triggered at the same time (by Timer 6).

Thus, during the ADC half-full callback, data can be written to the first half of the DAC buffer by the processor. It is safe to do because, at this time, the DMA system is writing to the DAC from the second half of the DAC buffer. The buffer-full callback is symmetrical with the half-full callback, so it is used to process data in the second half of the buffer.

The callback functions are also used to set the **ARD_D8_Pin** digital output before starting each buffer transfer and reset it afterwards. This will allow the timing of the code to be monitored, as was done in previous projects (Listing 5, lines 859, 864, 869 & 874).

Program execution results

Fig.13 shows the ADC and DAC signals from the oscilloscope for the DMA project with a similar setup to that used for Fig.8, except the sampling frequency is now 96kHz. The shorter step lengths can be observed on the DAC waveform. Another difference is the time offset of the waveforms – the samples are processed after the buffer has been half filled, which results in a delay from input to output.

Fig.14 shows the DAC output and the timing pulses from the oscilloscope for the DMA project. Unlike in the previous example, the timing pulses do not directly indicate the sampling. The pulses show when the processor is running code to transfer data between the ADC and DAC in the callback functions.

When the pulses are high (logic 1), transfer is taking place. At other times, the processor is effectively idle in this example – it is just looping in the main while loop, not doing anything specific. The behaviour of the timing pulses is illustrated in more detail in Fig.15.

There are two callback functions, so the timing pulses represent alternate callbacks – half full and full buffer. The length of the timing pulse is the time taken to run one of the callback functions (to process half the buffer). This is related to the processing speed, buffer size and the amount of code in the callback.

```
856 /* USER CODE BEGIN 4 */
857 void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)
858 {
859   HAL_GPIO_WritePin(GPIOB, ARD_D8_Pin, GPIO_PIN_SET);
860   for (int i= 0; i < HALF_BUF_SIZE; i++)
861   {
862     DAC_buffer[i] = ADC_buffer[i];
863   }
864   HAL_GPIO_WritePin(GPIOB, ARD_D8_Pin, GPIO_PIN_RESET);
865 }
866
867 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
868 {
869   HAL_GPIO_WritePin(GPIOB, ARD_D8_Pin, GPIO_PIN_SET);
870   for (int i= HALF_BUF_SIZE; i < BUF_SIZE; i++)
871   {
872     DAC_buffer[i] = ADC_buffer[i];
873   }
874   HAL_GPIO_WritePin(GPIOB, ARD_D8_Pin, GPIO_PIN_RESET);
875 }
876 /* USER CODE END 4 */
```

Listing 5: the DMA callback functions.

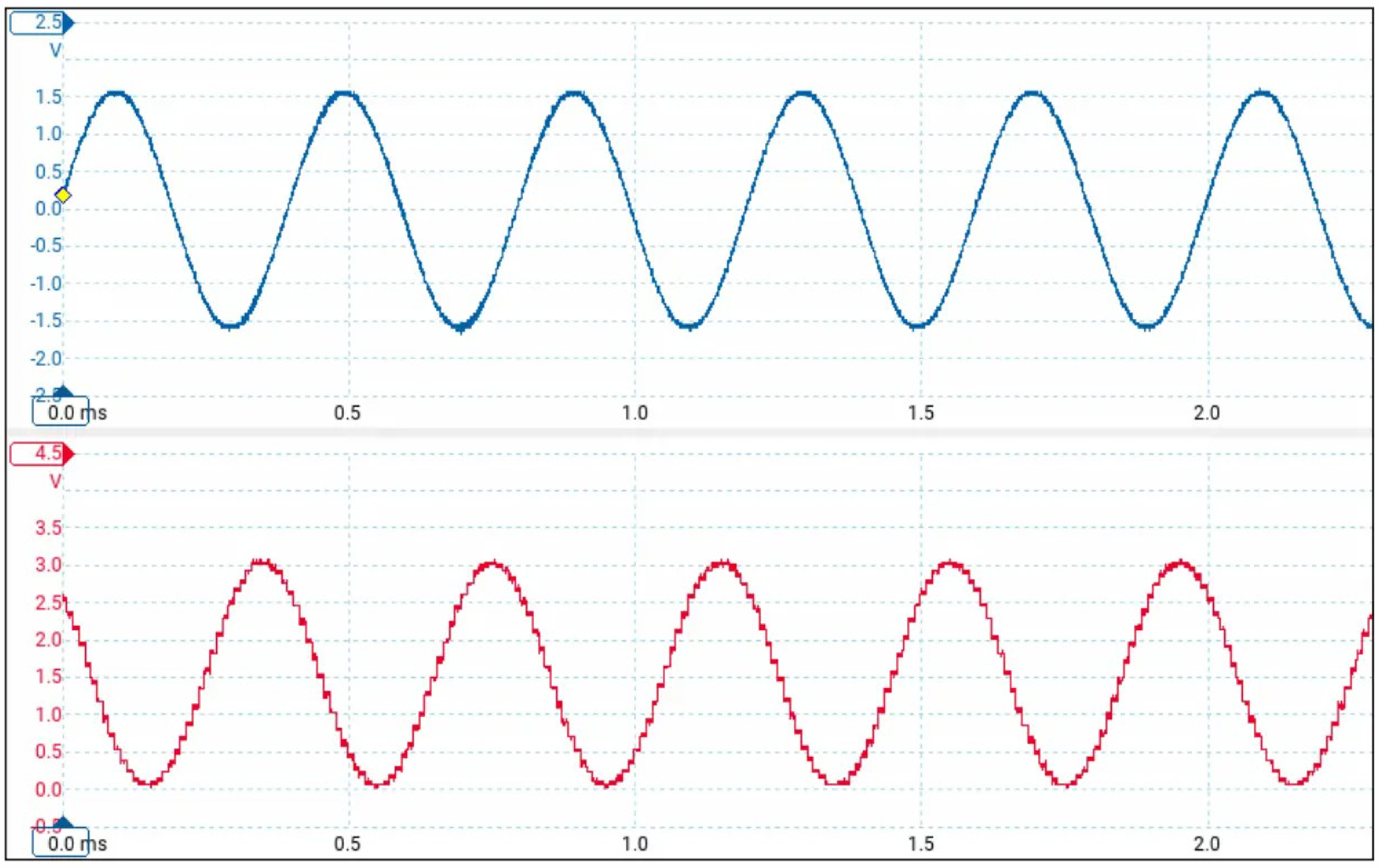


Fig.13: the ADC input and DAC output waveforms for the code in Listings 3-5.

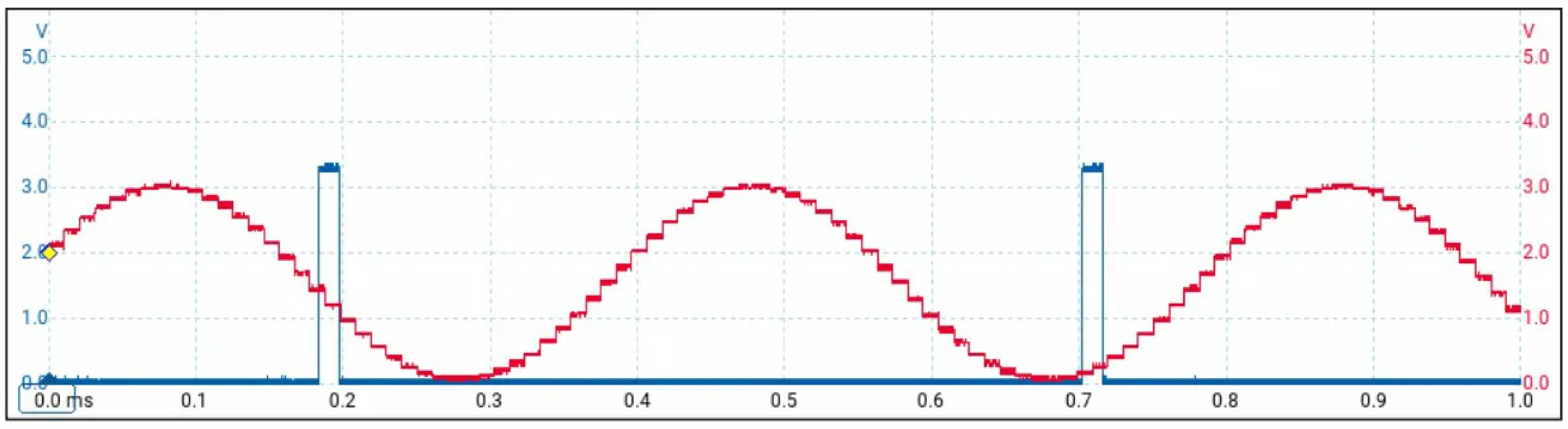


Fig.14: the timing pulses and DAC output for the code in Listings 3-5.

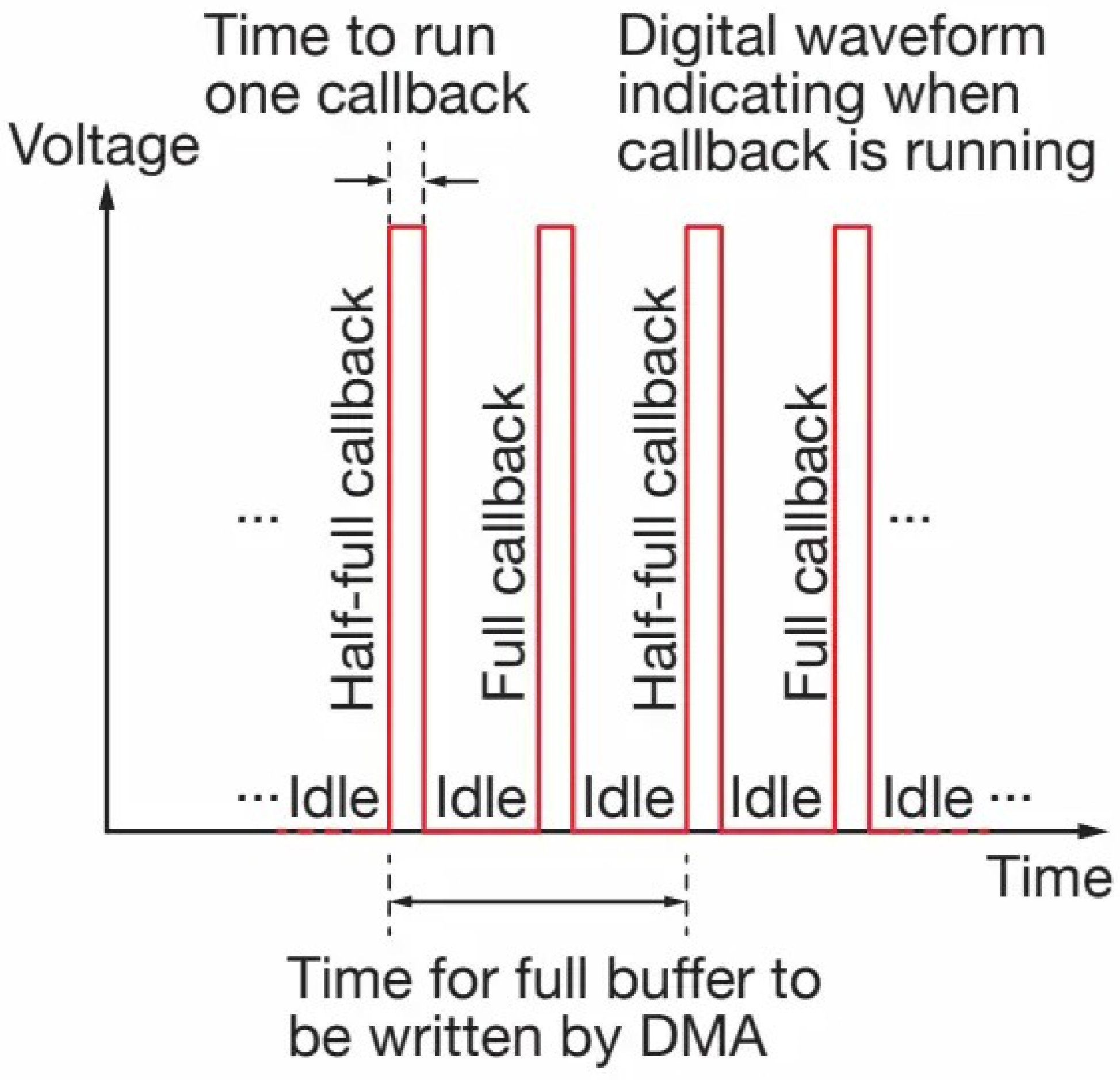
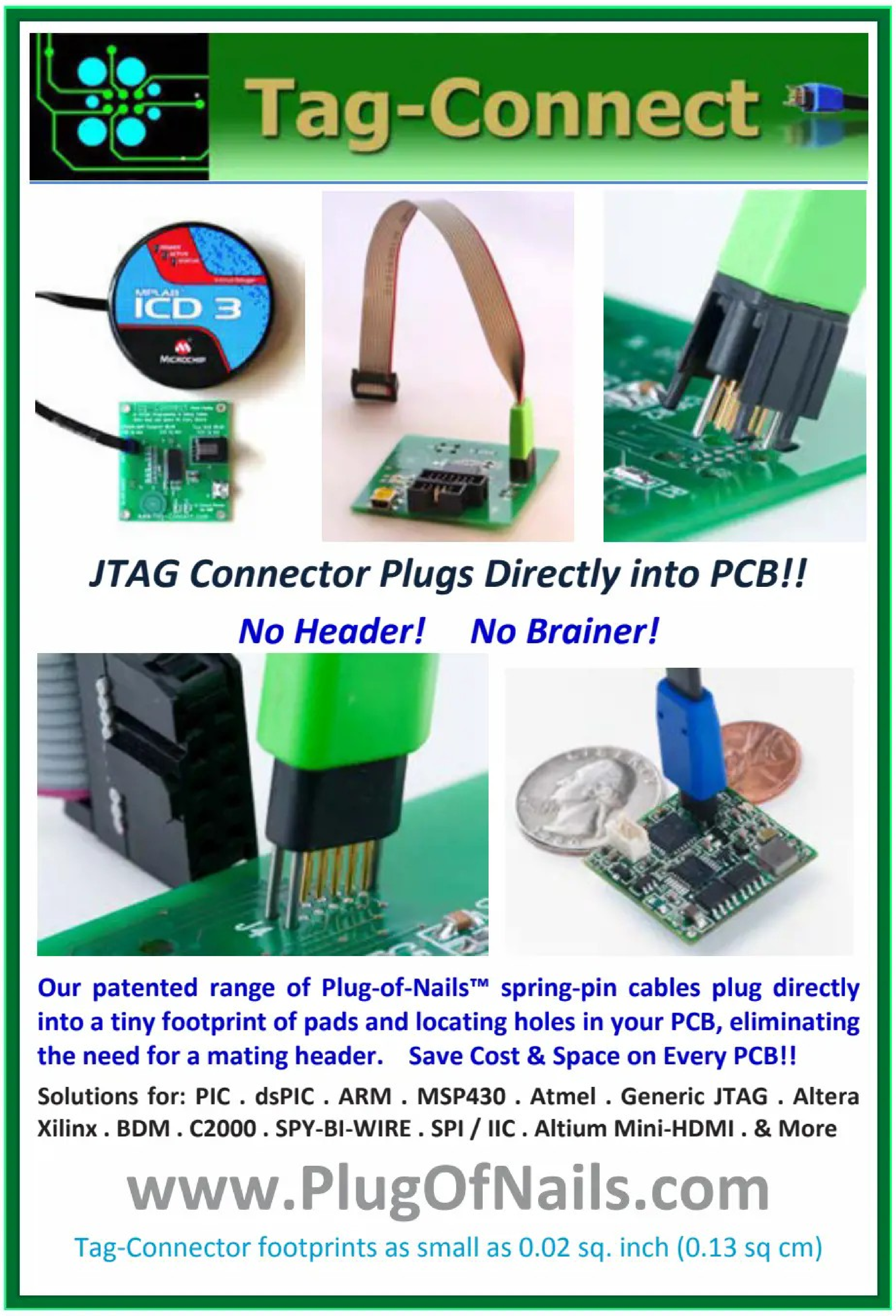

Time to run one callback
Digital waveform indicating when callback is running
Voltage
...
Half-full callback
Full callback
Half-full callback
Full callback
...
...Idle
Idle
Idle
Idle
Idle ...
Time
Time for full buffer to be written by DMA

Fig.15: timing pulses related to callback function execution and DMA transfers.

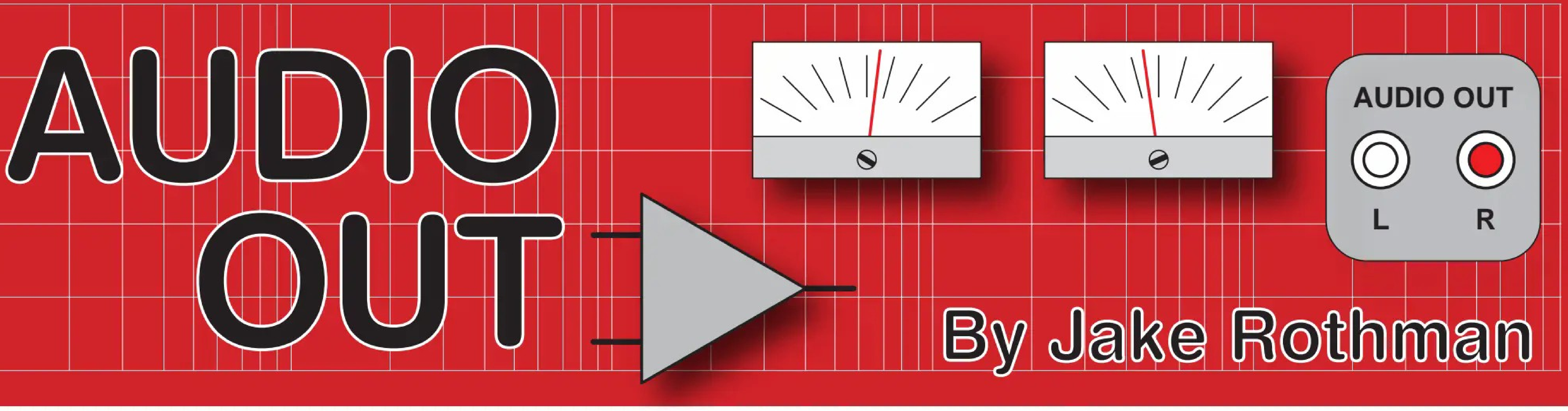
The time between alternate positive edges is the time taken for a full buffer to be written/read by the DMA system. This is related to the buffer size and sample rate. The total time taken to run two callback functions must be less than the time taken to write/read the full buffer.

As previously discussed, the DAC settling time of 3µs implies a maximum sampling rate of about 330kHz (the ADC is faster). Experiments show that the DMA processing code can run at sampling rates of 1MHz or more – compare this with the 60kHz limit for the first example.

At excess sampling rates, the DAC output looks OK, with a smooth slow input waveform, such as a relatively low frequency sinewave. However, there is reduced output amplitude. As this is faster than the DAC can operate correctly, it would be a poor choice in a real design. However, this result serves to illustrate the effectiveness of the DMA in speeding up signal processing. **PE**


Tag-Connect
JTAG Connector Plugs Directly into PCB!!
No Header! No Brainer!
Our patented range of Plug-of-Nails™ spring-pin cables plug directly into a tiny footprint of pads and locating holes in your PCB, eliminating the need for a mating header. Save Cost & Space on Every PCB!!
Solutions for: PIC . dsPIC . ARM . MSP430 . Atmel . Generic JTAG . Altera Xilinx . BDM . C2000 . SPY-BI-WIRE . SPI / IIC . Altium Mini-HDMI . & More
www.PlugOfNails.com
Tag-Connector footprints as small as 0.02 sq. inch (0.13 sq cm)

ENIGMA
COMPONENT SHOP
www.enigma-shop.com
Electronic Components for the Hobbyist!
Cable & Wire • Capacitors • Connectors • Filters
Kits & Modules • Hardware & Cooling
and much more...
READY TO BE REWARDED?
Join our loyalty scheme and **earn points** on every purchase you make which can be **used for discounts** in the future.
Read our reviews on ★ Trustpilot



AUDIO OUT

By Jake Rothman

Balance with Baxandall

Peter Baxandall was possibly Britain's greatest discrete analog audio circuit designer. He combined physics, magnetics, acoustics, music and electronics into one with great creative flair. He initially worked at the Royal Signals and Radar Establishment in Malvern.

The creator of stereo and the long-tailed pair, Alan Blumlein, was also based there, albeit before Baxandall, so they never met. I like to imagine what they would have come up with if they had.

Like Blumlein's work, a Baxandall circuit was always original, backed up with practical research, and cost-efficient. He is most famous for his bass-treble tone control, which has adorned almost every Hi-Fi amp since the 1950s. He won a competition with it and never got a royalty, just a $25 watch.

His series on power amplifier design in *Wireless World* in the late 1970s inspired Douglas Self's series in the 1990s. All engineers, like Newton, build on the shoulders of giants.

Jinxed!

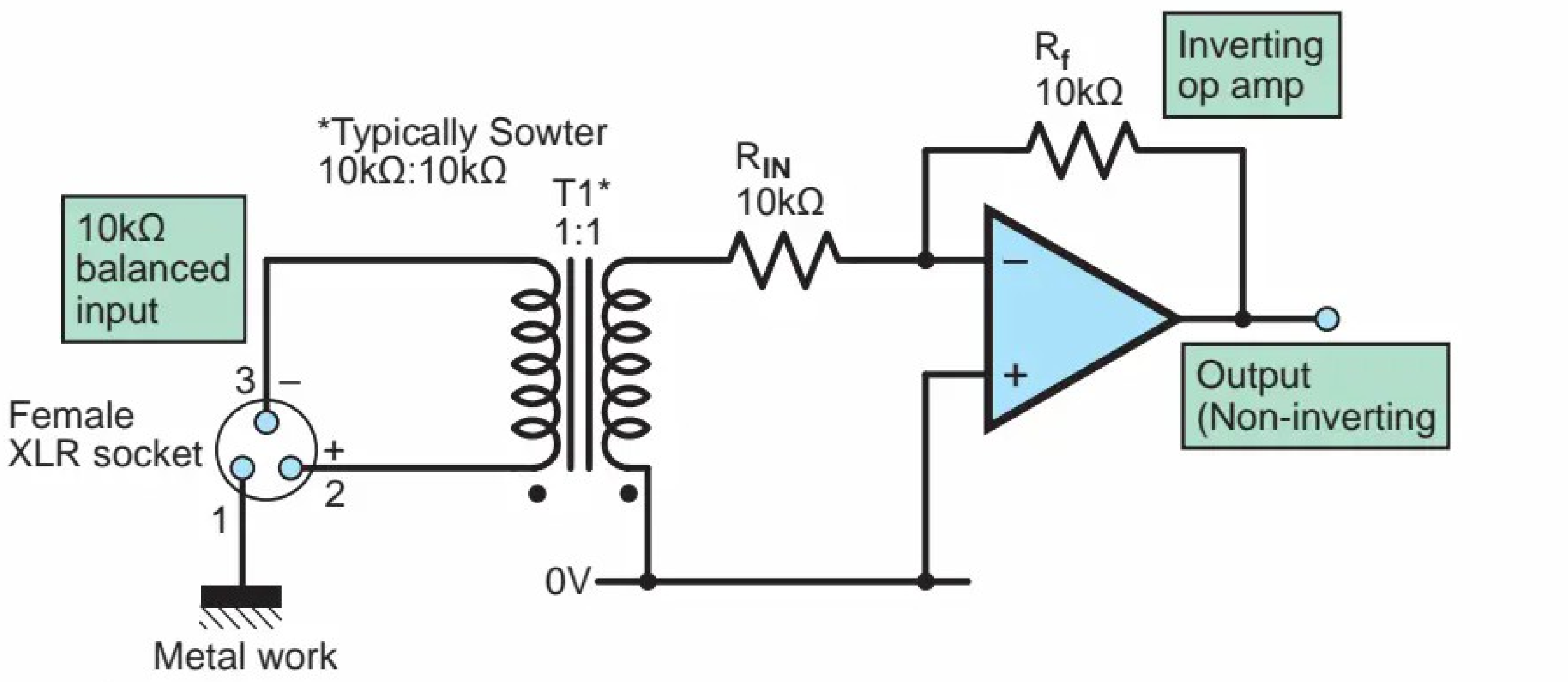
I wanted to interview Baxandall in Malvern for *Practical Electronics* when Mike Kenward was editor. Sadly, Baxandall died in 1995. Then I thought I'd try to interview John Linsley-Hood in Taunton instead, and he died too.

Not to give up, I went to Newport Radio Rally to interview the Reverend George Dobbs, designer of the Ladybird book transistor radio, but he cancelled due to ill health and died. I then went to interview John Birkett in Lincoln, owner of the longest-lived component shop in the UK. You can guess what happened. Am I cursed? You be the judge...

Anyway, back to stuff we can control. Following on from the previous investigations of obscure audio circuits, we'll look at Baxandall's balanced input circuit.

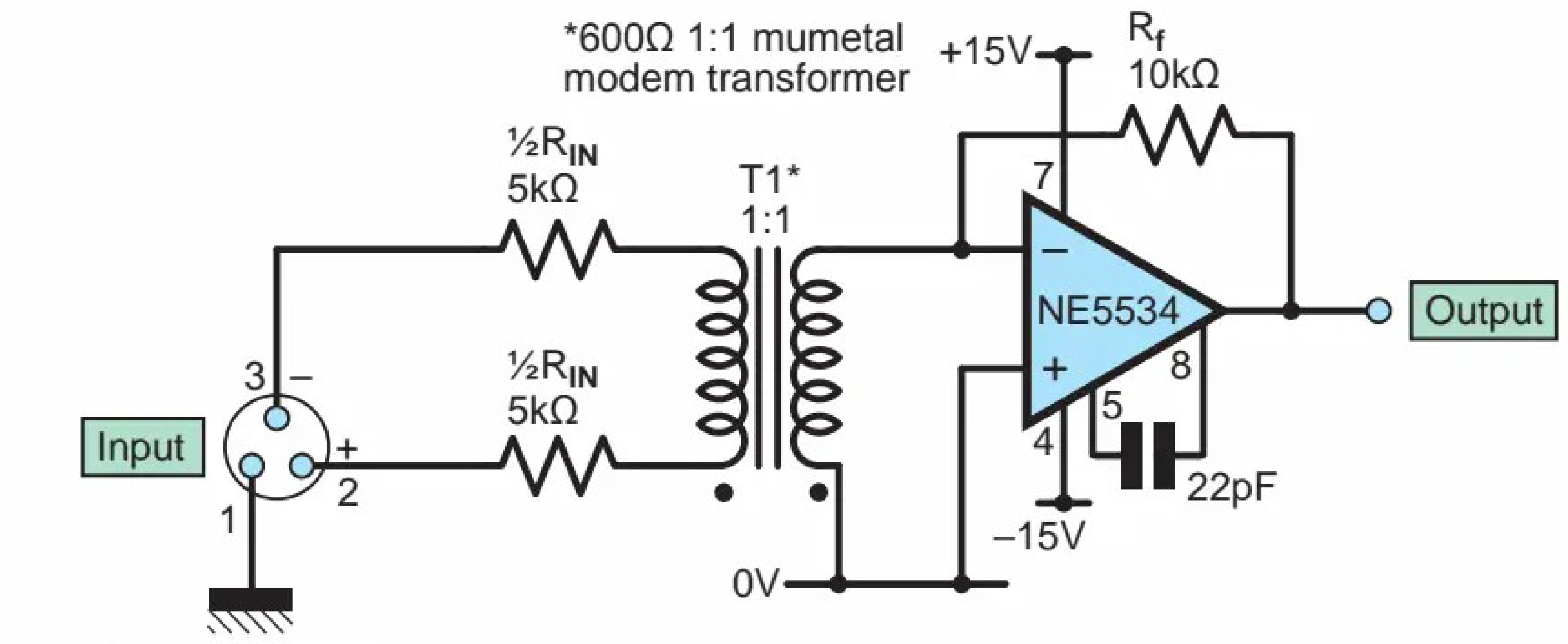
Transformer input

Normally, a differential op amp



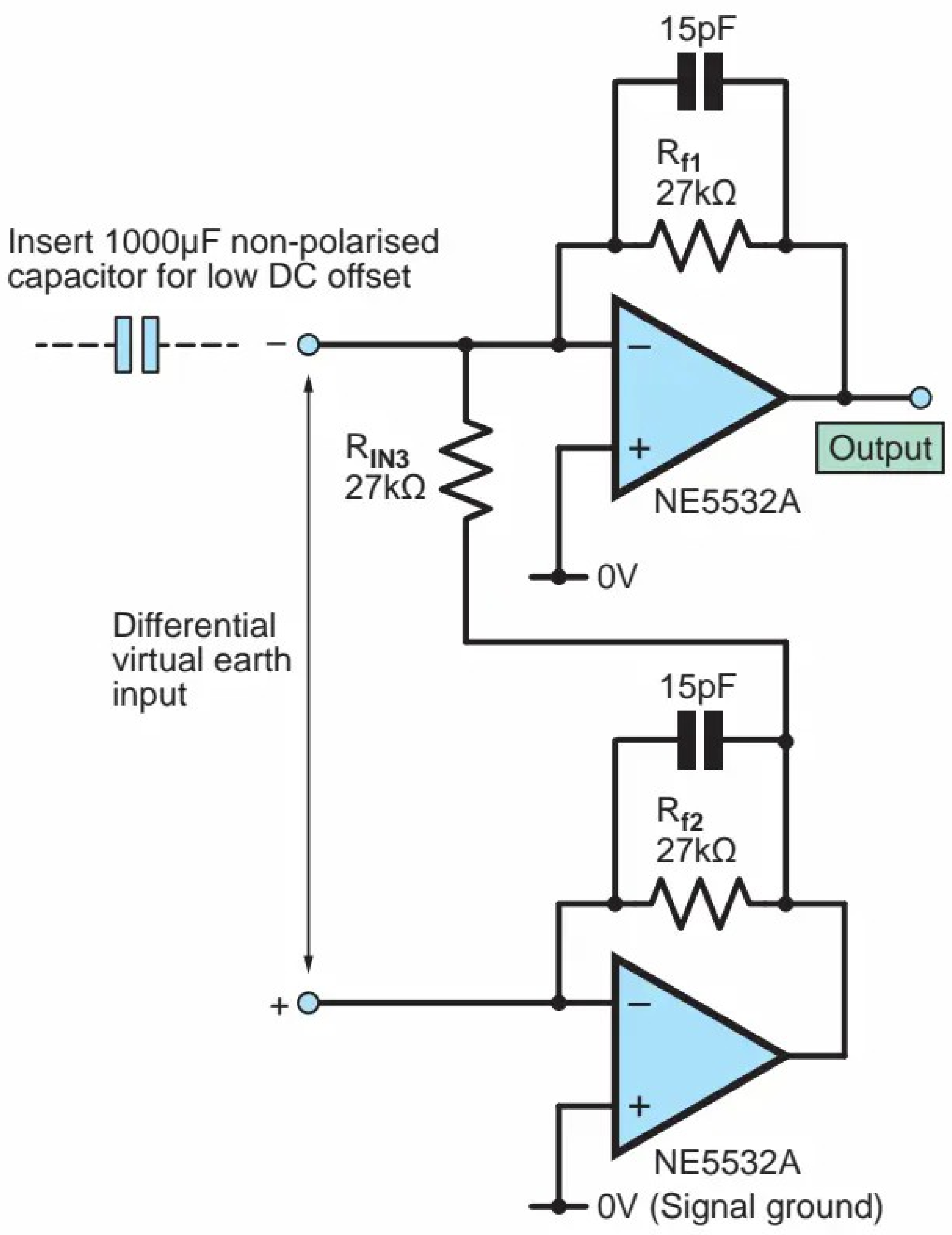
Rf
10kΩ
Inverting op amp
*Typically Sowter 10kΩ:10kΩ
T1*
1:1
RIN
10kΩ
10kΩ balanced input
Female XLR socket
3
–
+
2
1
Metal work
0V
Output (Non-inverting

Fig.1(a): an inverting amplifier with balanced input transfomer.



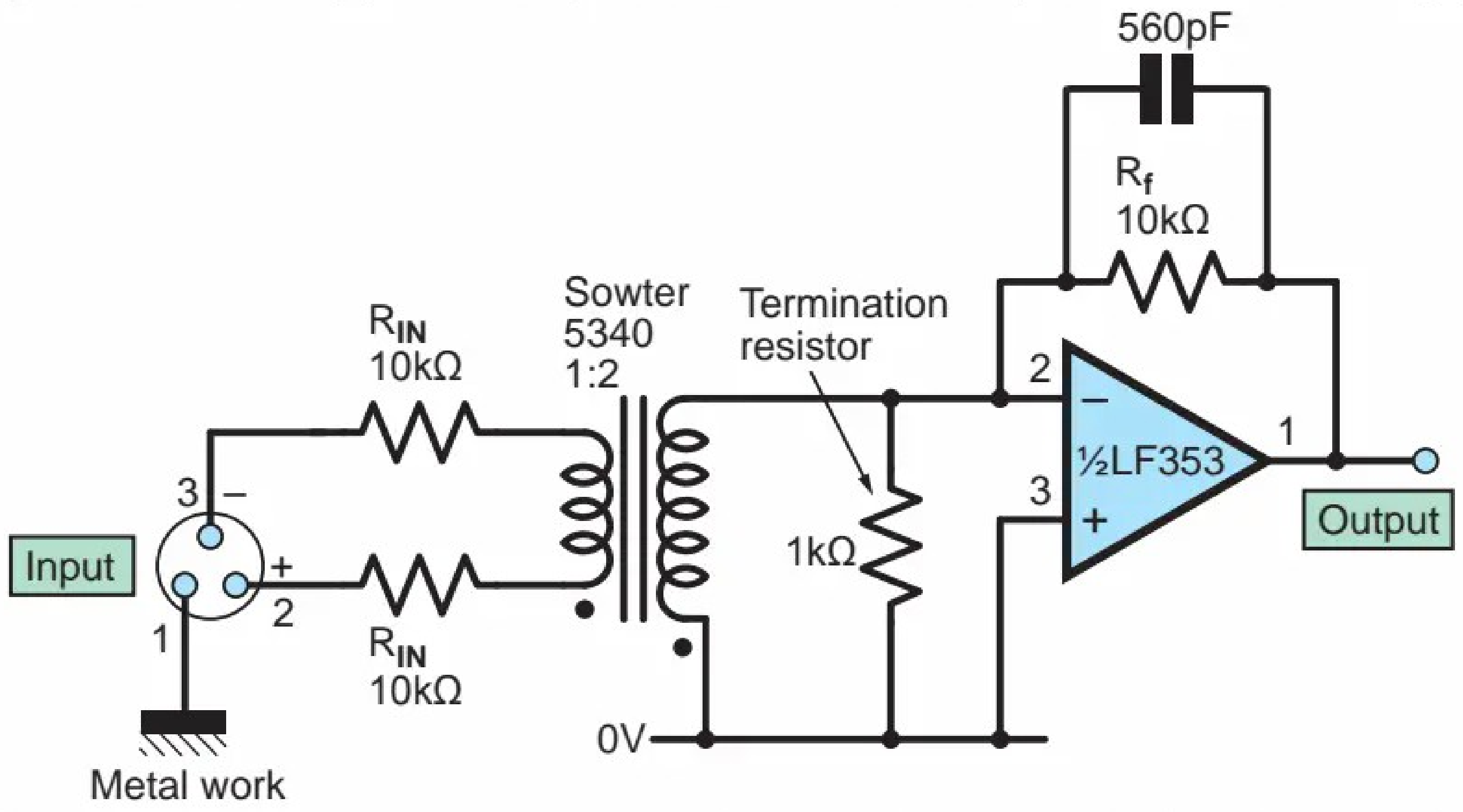
*600Ω 1:1 mumetal modem transformer
+15V
Rf
10kΩ
½RIN
5kΩ
T1*
1:1
7
NE5534
Output
½RIN
5kΩ
8
5
22pF
4
–15V
0V
Input
3
–
+
2
1

Fig.1(b): the Baxandall balanced input configuration. The input resistor(s) and input transformer positions are flipped.



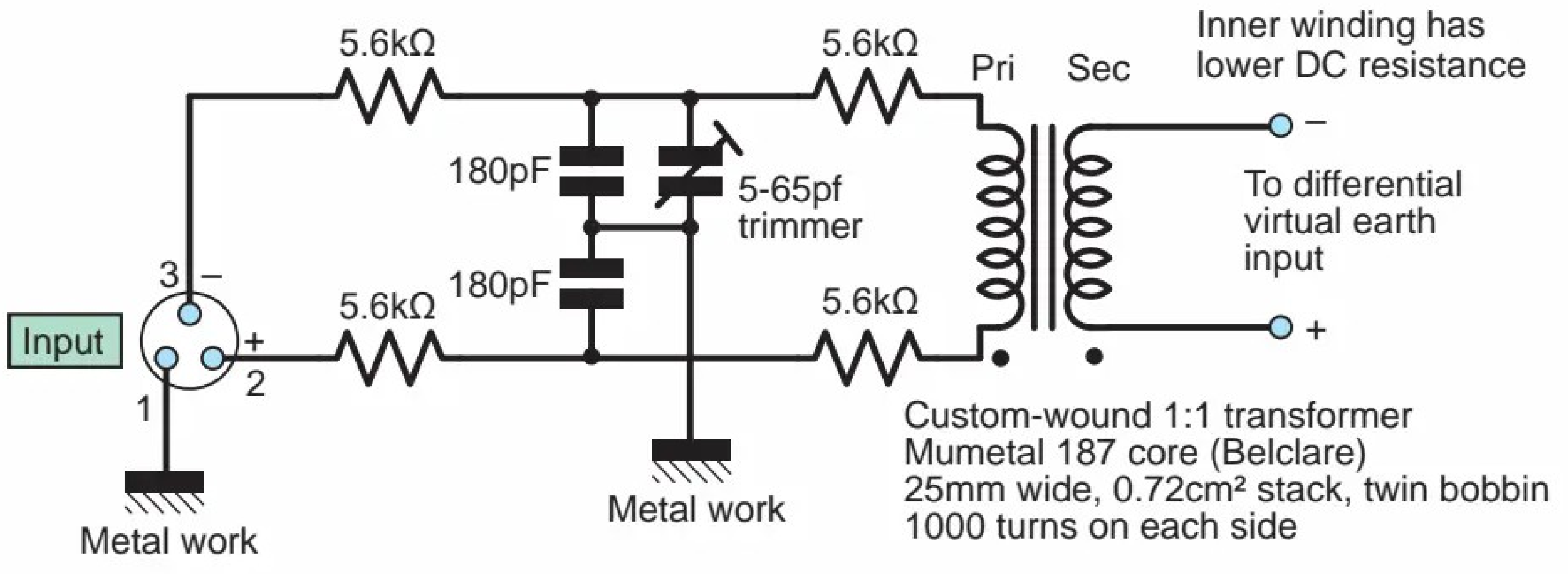
15pF
Rf1
27kΩ
Insert 1000µF non-polarised capacitor for low DC offset
–
RIN3
27kΩ
NE5532A
Output
0V
Differential virtual earth input
15pF
Rf2
27kΩ
+
NE5532A
0V (Signal ground)

Fig.3(a): a balanced virtual earth used in the full-spec Baxandall balanced input in the KEF KM1 active monitor speaker.



560pF
Rf
10kΩ
RIN
10kΩ
Sowter 5340 1:2
Termination resistor
2
3
1
½LF353
Output
1kΩ
Input
3
–
+
2
1
RIN
10kΩ
Metal work
0V

Fig.2: a commercial example of a Baxandall balanced line input (not very well optimised).



5.6kΩ
5.6kΩ
Pri
Sec
Inner winding has lower DC resistance
180pF
5-65pf trimmer
To differential virtual earth input
–
+
180pF
5.6kΩ
5.6kΩ
Input
3
–
+
2
1
Metal work
Metal work
Custom-wound 1:1 transformer
Mumetal 187 core (Belclare)
25mm wide, 0.72cm² stack, twin bobbin
1000 turns on each side

Fig.3(b): the transformer stage feeding the above amplifier. Most 1:1 600Ω to 2kΩ small audio transformers will work well.

circuit would be used for a balanced input because it is linear and low in cost. A transformer input, however, is fully isolated from Earth (floating) and has a huge common mode rejection ratio (CMRR). This means it is very useful in areas of high interference, such as broadcast studios and electric transport.

The common-mode rejection ratio is the amount by which a common signal to both inputs (such as mains hum) is attenuated at the output of the circuit. So with a 100dB CMRR and a 100mV unwanted common-mode signal, only 1µV of that 100mV will make it through to the output.

I remember a situation with a ship-based pirate radio station, where the RF induced a voltage in every bit of metal, and an op amp balanced input would have been saturated. The 'old-fashioned' transformer inputs on the desk worked fine, even with 50V of common mode signal on the positive and negative inputs.

There are expensive problems with audio balancing transformers; it typically costs £12-40 for one that has good frequency response and low distortion. The good news is that with Baxandall's circuit, a cheap transformer can give excellent results.

Inverting the inverted

Sometimes one of the simplest circuit improvements can simply be to swap over the positions of two components. This was the key to the success of Baxandall's design, which was based on an inverting op amp circuit.

Normally, the input balancing transformer feeds the input resistor R_{IN}, as in Fig.1(a). Having the transformer feed the inverting input directly and putting the input resistor first, as in Fig.1(b), makes the transformer more linear. This gives a much better frequency response and lower distortion.

This is because the inverting input is a virtual ground, meaning there is almost no voltage across the secondary. Since a transformer reflects what is happening onto the other side, there is no voltage across the primary either; all the input voltage is dropped across R_{IN}.

This setup makes the transformer operate in current mode, and because this current is set by the input resistor, which is inherently linear, the currents in the transformer windings are forced to be linear. With a transformer operating in normal voltage mode, saturation and hysteresis effects are much more pronounced.

Practical circuits

To optimise the CMRR, R_{IN} is split into two individual resistors, which positions the transformer's capacitance symmetrically, helping the balance at high frequencies. The first example of Baxandall's circuit I saw was in a line input for a Sonifex call box used in broadcast systems, shown in Fig.2. I had to service one in the early eighties.

This circuit strangely used an expensive Sowter transformer (similar to the one in Photo 1) with a noisy JFET op amp. It also had an odd termination resistor, R_T, to ensure stability, which increased the noise gain. It would have worked better if it had had a small capacitor in series with R_T, making it into a Zobel network.

The circuit reached a new level of sophistication in the input stage for KEF's KM1 active studio monitor, for which Baxandall designed the electronics. It was made for the BBC and Abbey Road studios. A differential virtual earth amplifier was used, as in Fig.3(a), which gave another layer of common-mode signal rejection.

The transformer input stage shown in Fig.3(b) also split the input resistors to include an HF trimming network.

A big electrolytic capacitor was included to minimise the DC offset, which was otherwise around 300mV because of the high input bias currents of the NE5532 op amps. This caused a low-frequency (LF) hump of 5dB at 0.8Hz by resonating with the secondary inductance. This could be fixed by putting in an LF step network in series with the capacitor, or with the offset removed entirely using a servo.

There is another variation of the circuit (by Barry Blesser, on p297 of the *Journal of the Audio Engineering Society*, May 1972) where the feedback resistor is fed into a third winding to cancel the flux, which shrinks the transformer even more.

Winding your own

Baxandall was highly adept at designing signal transformers, a lost skill today. He used to give full winding details for many of his transformer-based designs, such as his *Wireless World* March 1957 Inexpensive High-Quality Amplifier.

This input stage was no exception, and the details are given in Fig.3(b). Since the primary has series resistors, it can be wound with very thin wire. The secondary should be wound with thicker wire on the inside winding for lower resistance, giving lower noise.

Sadly, one can't get Belclere transformer kits any more, which included the laminations and bobbin. But I have found there are Mumetal 600Ω 1:1 modem isolation transformers available inexpensively on the surplus

SOWTER
TYPE

Photo 1: a typical expensive input transformer in a Mumetal can.

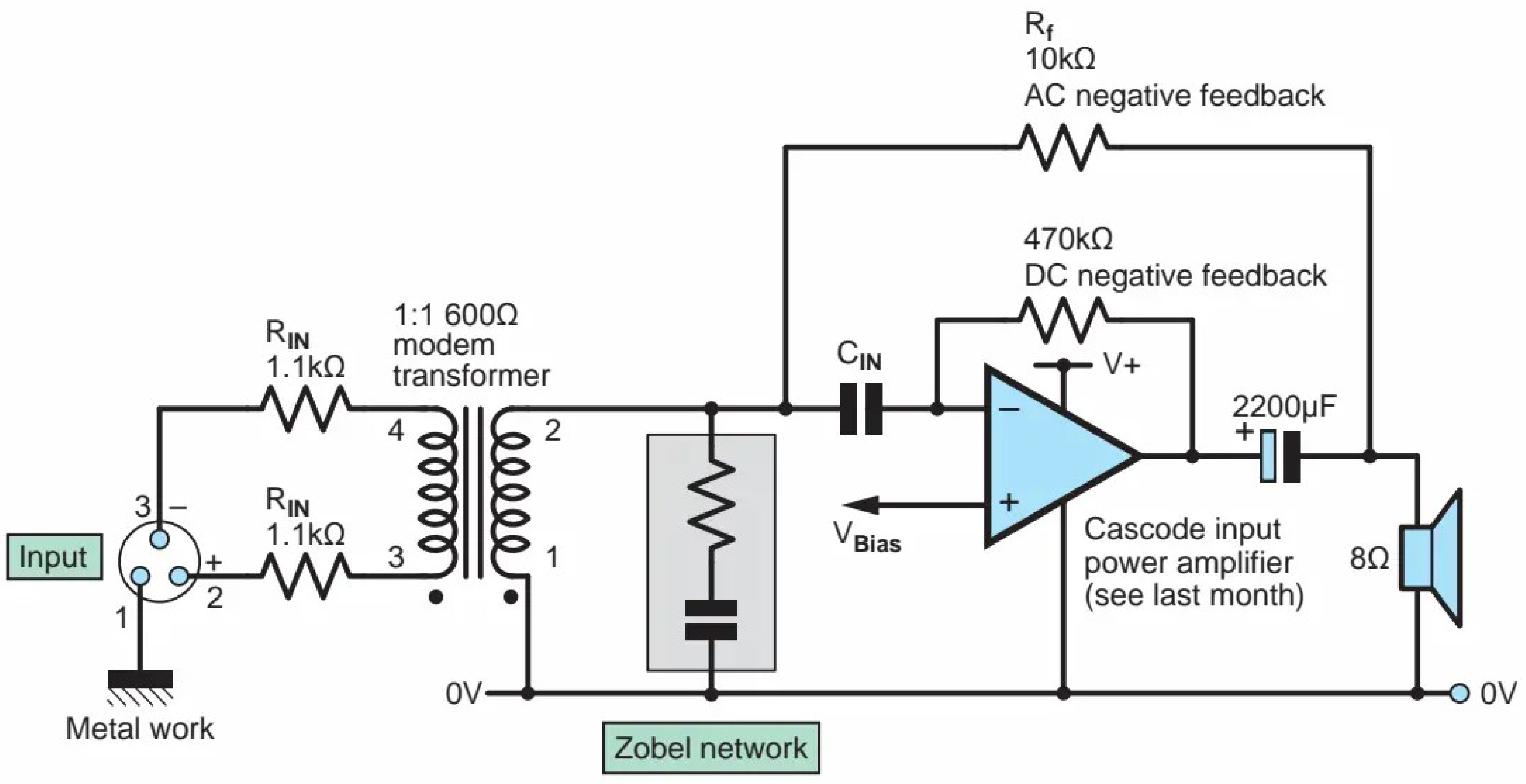

R$_f$
10kΩ
AC negative feedback
470kΩ
DC negative feedback
R$_{IN}$
1.1kΩ
1:1 600Ω
modem
transformer
C$_{IN}$
V+
2200µF
+
4
2
3
1
3
–
R$_{IN}$
1.1kΩ
Input
+
2
1
V$_{Bias}$
Cascode input
power amplifier
(see last month)
8Ω
0V
0V
Metal work
Zobel network

Fig.4(a): two weird circuits joined into one. Adding the Baxandall balanced input to last months complementary cascode amplifier. Unfortunately, the inductance of the transformer resonates with the capacitor C_{IN} and the AC feedback path.

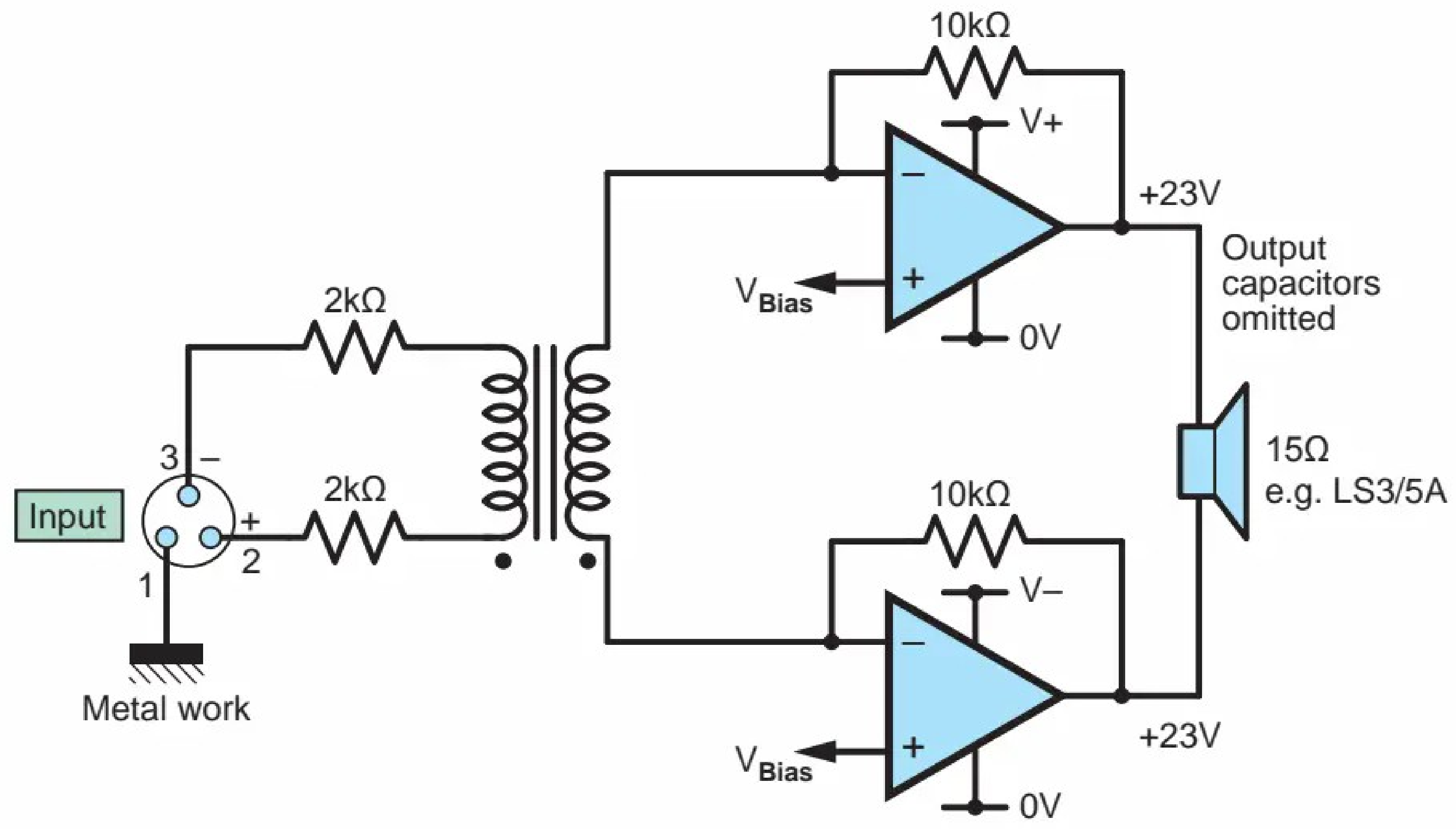

10kΩ
V+
+23V
Output
capacitors
omitted
V$_{Bias}$
0V
2kΩ
3
–
Input
+
2
1
2kΩ
Metal work
15Ω
e.g. LS3/5A
10kΩ
V–
V$_{Bias}$
+23V
0V

Fig.4(b): using the Baxandall circuit with two complementary cascode amplifiers to make a more powerful bridge set-up. The distortion should, in theory, be lower than one amplifier.

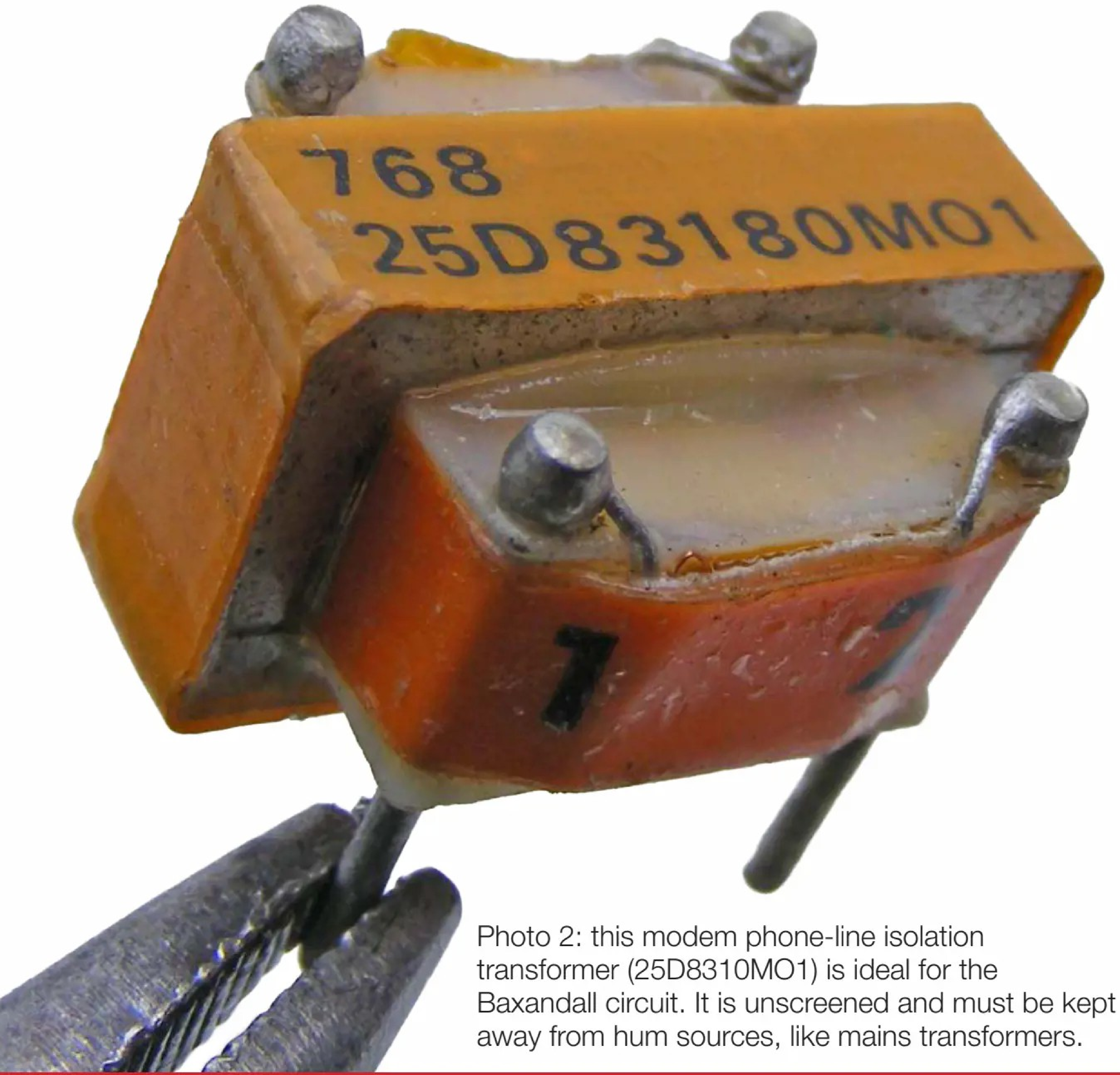

768
25D83180MO1
1

Photo 2: this modem phone-line isolation transformer (25D8310MO1) is ideal for the Baxandall circuit. It is unscreened and must be kept away from hum sources, like mains transformers.

market that do the job (see Photo 2). I bought 200 of them just so you can have some as well! Note that the inner winding is pins 1 and 2.

Ferrite pot cores are not normally used for audio signals, but the Baxandall circuit compensates well for their shortcomings. If I was designing a balanced input for, say, a railway public address system, where maximum fidelity was less important, I would use them. They are well-shielded with the winding enclosed, and don't need a screening can like laminated transformers.

It is well worth reading Baxandall's chapter, Microphone amplifiers and transformers, in the *Microphone Engineering Handbook* by Michael Gayford, published by Focal Press.

Performance

Using the circuit shown in Fig.1(b) with $R_f = R_{IN}$ =10kΩ, using a NE5534 op amp and the modem transformer at 0dB (line level), the total harmonic distortion plus noise (THD+N) measured 0.0018% between 100Hz and 20kHz. It rose gently to 0.028% at 20Hz and then was flat down to 5Hz.

Using the transformer in the conventional way gave over 5% THD+N at 20Hz due to saturation, falling to 0.01% at 1kHz.

The Baxandall circuit basically reduced the core flux density to a minimum at LF. The >100Hz THD+N reduced to 0.001% when the resistors were reduced to 5.1kΩ because of the lower Johnson (thermal) noise – see Fig.5.

Note that to ensure HF stability, R_f needs a 47pF phase-lead capacitor across it, and the NE5534 needs a 22pF compensation capacitor between pins 5 and 8.

Putting it all together

The complementary cascode power amp described last month, being an inverting configuration, could combine nicely with the Baxandall balanced input, as shown in Fig.4(a). There is an interaction between the transformer's inductance and the loudspeaker output and input capacitor (Cin), causing a hump in the LF response that needs resolving, though.

Another interesting project would be to make a bridged version to more than double the power output, like in Fig.4(b), using the balanced drive from the transformer. Since the complementary cascode amplifier is asymmetrical, the even harmonic distortion components should cancel out, dropping the THD.

In theory, the output capacitors could be left off (avoiding the LF hump problem) since both output pins are sitting at the same voltage, but I would be worried about transient DC currents flowing through the speaker. There's only one way to find out, and that's to build it. I'll do that as soon as I have some proper PCB designs ready to go.

Analog audio design is like cooking; it's how you combine the elements and their interactions that matter. Very few circuits are truly new; just different combinations.

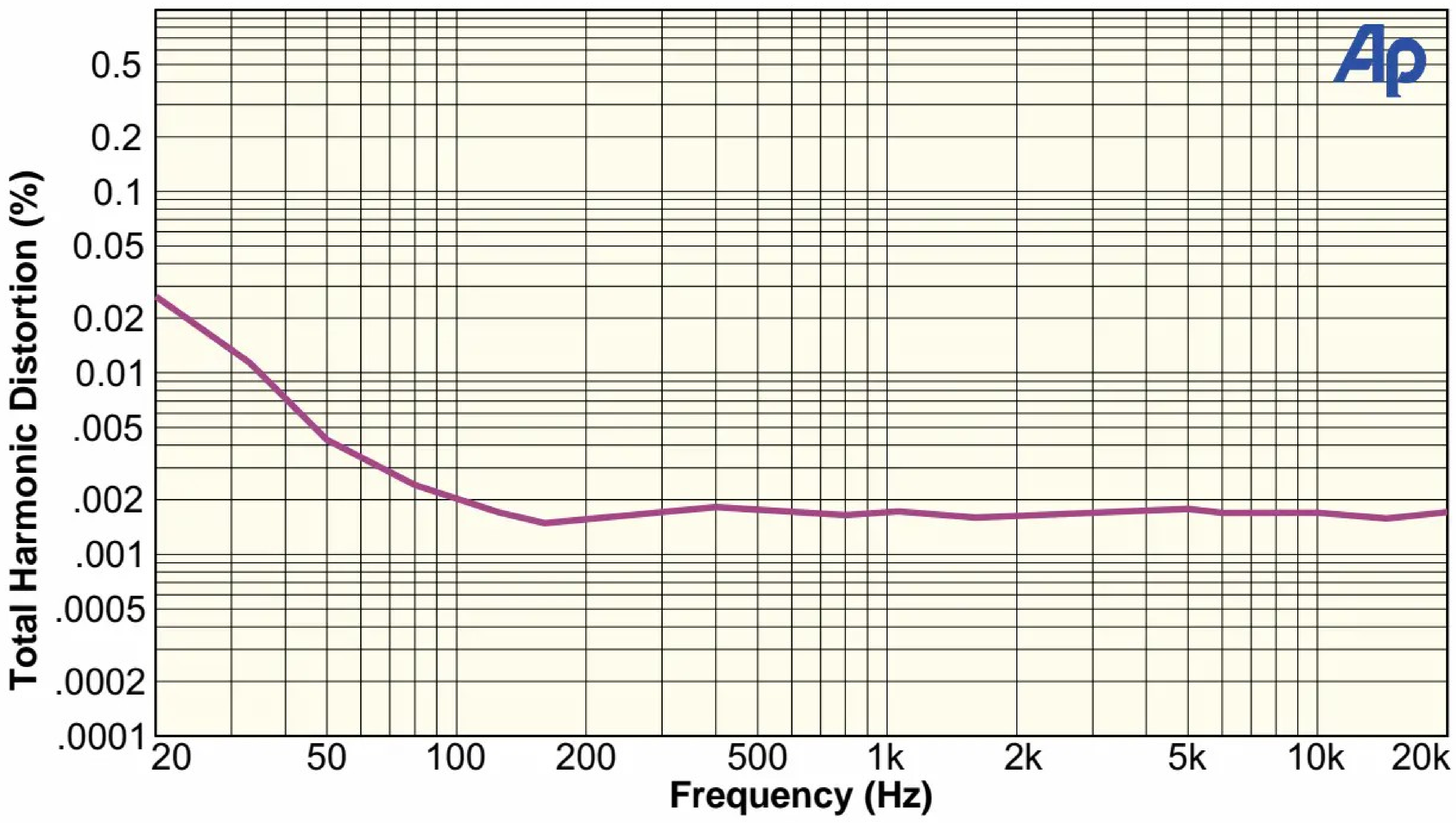


Fig.5: the Baxandall balanced input distortion plot with a modem transformer (768 25D8310MO1) and NE5534 op amp at full output (27V peak-to-peak into 600Ω).

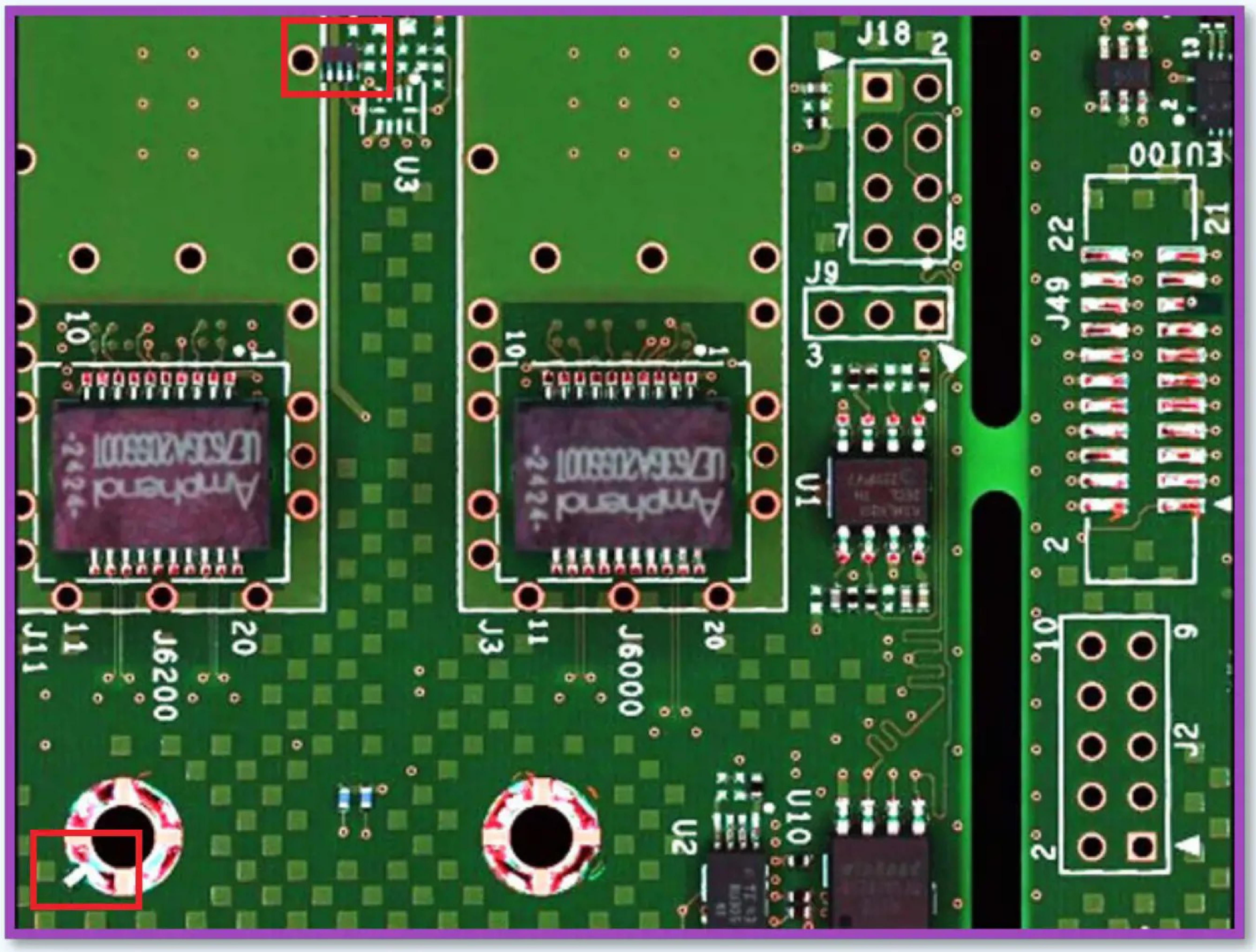
Minor erratum

In last month's article, I mentioned that the Bailey and Quad 303 amplifiers used rare Motorola transistors that had minimal Early effect. I should have written that they were RCA transistors instead. **PE**

Techno Talk puzzle

This question was posed on the bottom of page 5, asking readers to spot the difference between two images of circuit boards, one a reference and the other a productiton part, which was used as a test for an image difference analysis system.

The system detected an extra part, which was added as a test (upper red box). It also detected foreign object debris, which could become a problem at some point in the future (lower red box).



FIND ALL YOUR ELECTRONIC COMPONENTS IN ONE PLACE

BASIC MICRO UK
ELECTRONICS COMPONENT SUPPLIER
WWW.BASICMICRO.CO.UK

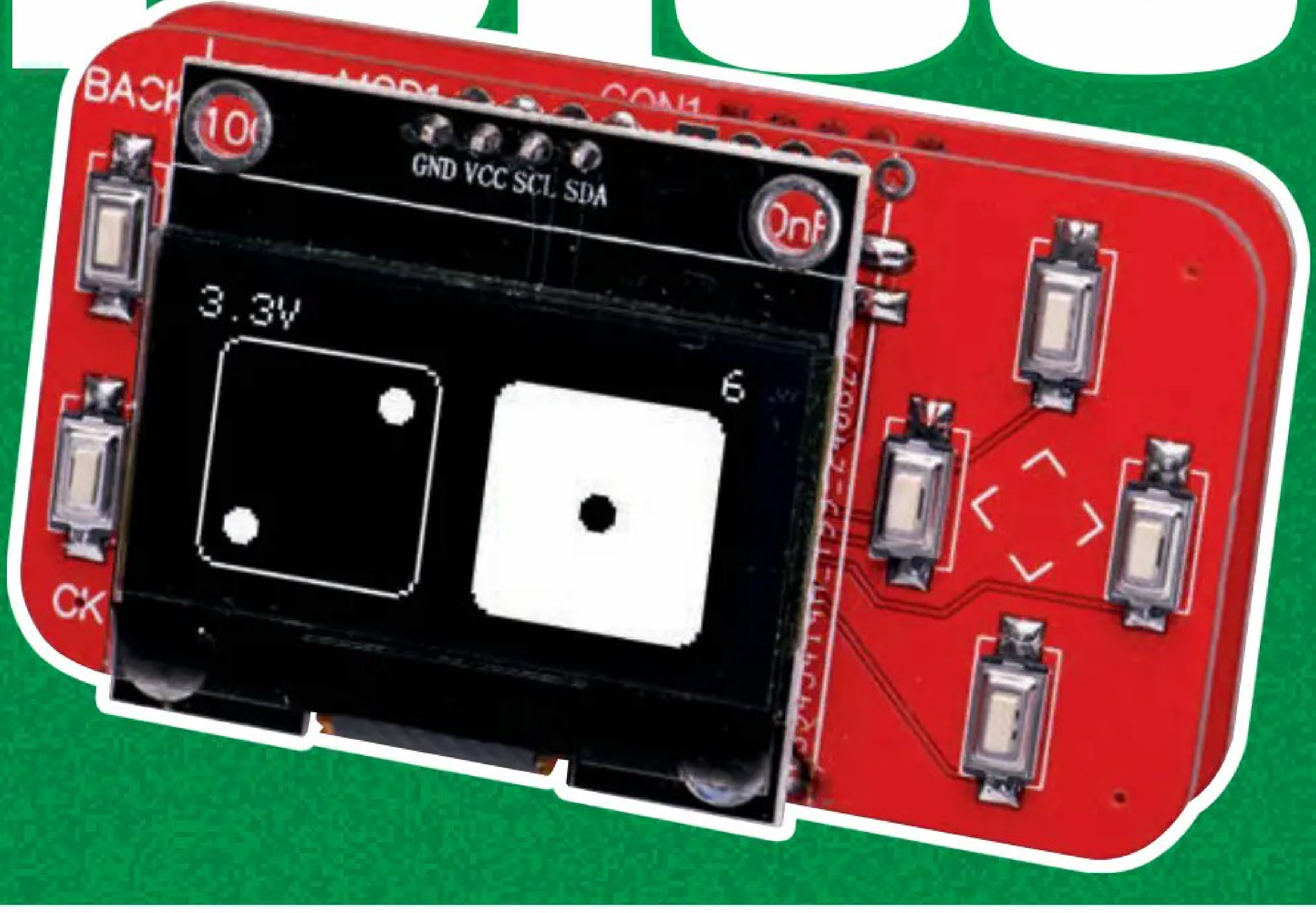
High-quality, genuine parts

01623 720730
Available from 9AM - 5PM Mon-Fri

FlexiDice

Project by Tim Blythman

We've published several dice projects over the years but all have been traditional six-sided affairs. Other types of dice are used for various games and activities. This dice project can emulate dice with up to 100 faces, coin tossing and other random events.



Not long after publishing our Dual Mini LED Dice project in the June 2025 issue, we thought it'd be nice to have a more configurable 'digital dice' design.

While some dice-rolling designs use a microcontroller, most of our electronic dice designs, including the latest iteration, have used straightforward digital logic to implement the throw and display of the dice.

One thing these designs all have in common is that they only emulate dice with six faces. The FlexiDice uses a microcontroller and an OLED display, so it can emulate just about any number of faces. We have chosen to allow up to 100, as that is the highest number of faces we have found on a real-life die.

The FlexiDice is compact and handheld, running from a coin cell. The board uses surface-mounting components, but they are M3216 (3.2 × 1.6mm) passives, SOIC ICs and several larger parts, so it is not too difficult to build.

A lot of games need only one or two six-sided dice, but many games use other types of dice or larger numbers. For example, the Dungeons and Dragons role-playing game (like many other role-playing games) uses dice with four, eight, 10, 12 and 20 sides. That includes 10-sided dice marked in tens so that its result can be combined with a regular 10-sided die to produce one of 100 different values.

Dice with 100 sides exist but can be impractical. Our FlexiDice can give rolls up to 100 but won't roll off the table. *Source:* https://w.wiki/AjL4

Dice with 100 sides exist, although their near-spherical shape makes them impractical to use because they do not stop rolling as quickly as smaller dice. One such example is shown below.

Most of the other dice mentioned (with four, six, eight, 12 and 20 sides) are regular polyhedrons, so they are symmetrical with regard to their faces, while the 100-sided dice are not. Asymmetrical dice like the D100 may not show each face with equal probability.

Those familiar with Dungeons and Dragons will also know the abbreviations used for various combinations of dice rolls. A roll of a single six-sided die would be abbreviated as "D6", while the roll of two six-sided dice (as in games like Monopoly) is "2D6". We also use this terminology with the FlexiDice.

Other games of chance use playing cards or coins to give a random result. The FlexiDice can emulate flipping two coins and shows images that resemble those found on an Australian penny, as is traditional in the historic Australian game Two-Up.

The FlexiDice can also display playing cards and imitate decks with between zero and six Jokers. While most people will be familiar with decks with two Jokers (and thus 54 cards), the German game Zwickern has six Jokers in a 58-card deck. The Jokers are removed for many games (hence the zero option).

The odds

A microcontroller is designed to provide a deterministic outcome; the same input should result in the same output, so we need a way to inject some randomness into its behaviour.

Of course, having a truly random outcome is the essence of dice, so we must ensure our means of generating random numbers is fair and not predictable. That the result is fair means that each outcome has a reasonably equal chance.

For a result to be not predictable, results must be independent of each other. The most common method of generating randomness in our earlier

Features & Specifications

- Compact, handheld device
- Runs from one 3V lithium coin cell
- Operates down to 2.4V
- Auto power-down with <1µA sleep current
- Fun mini-game console form factor
- Graphic display for easy viewing
- Shake-to-roll vibration sensor
- Hardware-based random number generator
- Ten configurable roll presets plus numerous user presets

Roll Types

Dice with two to nine pips

Numeric dice from two to 100

Random card pick with 0-6 Jokers

Coin toss (heads or tails)

dice projects is to use the variability in user input to randomise the result.

That usually involves the user pressing a button or switch to activate the roll. The exact time the button is held down is used as the random element. As long as the hardware can cycle through the states fast enough, the user cannot influence the outcome, and the result is random.

As we found during the development of the Dual Mini LED Dice, that is not always sufficient. In that case, we found that if the values of two critical timing capacitors were similar, the two halves of the circuit would interact and synchronise, resulting in the two dice often having the same result.

The result could still be fair, but it was also predictable, which is undesirable. Fortunately, using two different values of timing capacitor was enough to overcome this with the Dual Mini LED Dice.

The FlexiDice measures user input, but that is not the only source of randomness. We investigated several different noise (true random data) sources to see what would be suitable for this project.

Avalanche diodes

One of the better-known random noise sources is the breakdown behaviour of a reverse-biased PN junction (avalanche breakdown). A high enough voltage applied to the junction reaches a critical point that causes a rapid and unpredictable increase in current; an avalanche. In situations where the current is sustained, this can cause heating and damage.

When avalanche breakdown is used for noise generation, a series resistor limits the current to avoid damage to the junction and allows it to recover and experience further random events. Many such noise sources make use of the emitter-base junction of a transistor, with the collector being left unconnected.

The necessary breakdown behaviour requires at least 6V, and the noise level is quite low, so substantial amplification is needed. These factors conspire to make such a noise source difficult to operate from a coin cell; hence, we looked at other options.

Pseudo-random (LFSR)

The White Noise Generator project in the November 2019 issue used a different noise generation method. In this case, the output is known as ‘pseudo-random’ since it is not truly random but generated by a deterministic process.

Since they are deterministic, many pseudo-random processes can be proven to be fair and uncorrelated, but as the name suggests, they are not truly random.

There are many types of pseudo-random noise sources, but one of the simplest to implement is the linear feedback shift register (LFSR). This is a shift register with its input being a linear combination of some of its outputs, usually by XORing some carefully chosen register bits.

The FlexiDice implements a 31-bit LFSR that works in much the same fashion as that in the Digital White Noise Generator from 2019, although we do not use it as the primary source of randomness.

This 31-bit LFSR cycles through nearly all 31-bit states (and thus over two billion 31-bit numbers) and so takes very many cycles to repeat. The all-zero state is the only state that is avoided since it results in the LFSR being stuck in that state.

Since the LFSR's future state can be known from its current state, it can be very predictable. For example, we tested the FlexiDice using the LFSR as its only input and, unsurprisingly, the dice rolls were identical every time it was powered on.

Noise multiplier

The circuit we have implemented is known by various names, but “noise multiplier” seems the most appropriate. As the name suggests, the circuit amplifies noise from all sources, so even power supply noise enhances its operation.

Unlike an avalanche diode, only modest amplification is needed, and one of the outputs is digital in nature, allowing it to be easily fed to a microcontroller.

Consider the sub-circuit shown in Fig.1. The left-hand op amp is wired as a comparator, with its inverting input connected to a half-rail reference generated by a divider. A signal is applied at V_{IN}; if it is more than half of the supply voltage, BIT_OUT is high; otherwise, it is low.

The right-hand op amp is configured to have a gain of two, with the inverting input referenced to BIT_OUT. In other words, V_{OUT} will be double V_{IN} minus BIT_OUT. Scope 1 shows these values as V_{IN} is simulated being swept from 0V to V_{CC}.

What the circuit is doing is doubling the voltage (including any noise present); hence the term noise multiplier. By maintaining its output between the

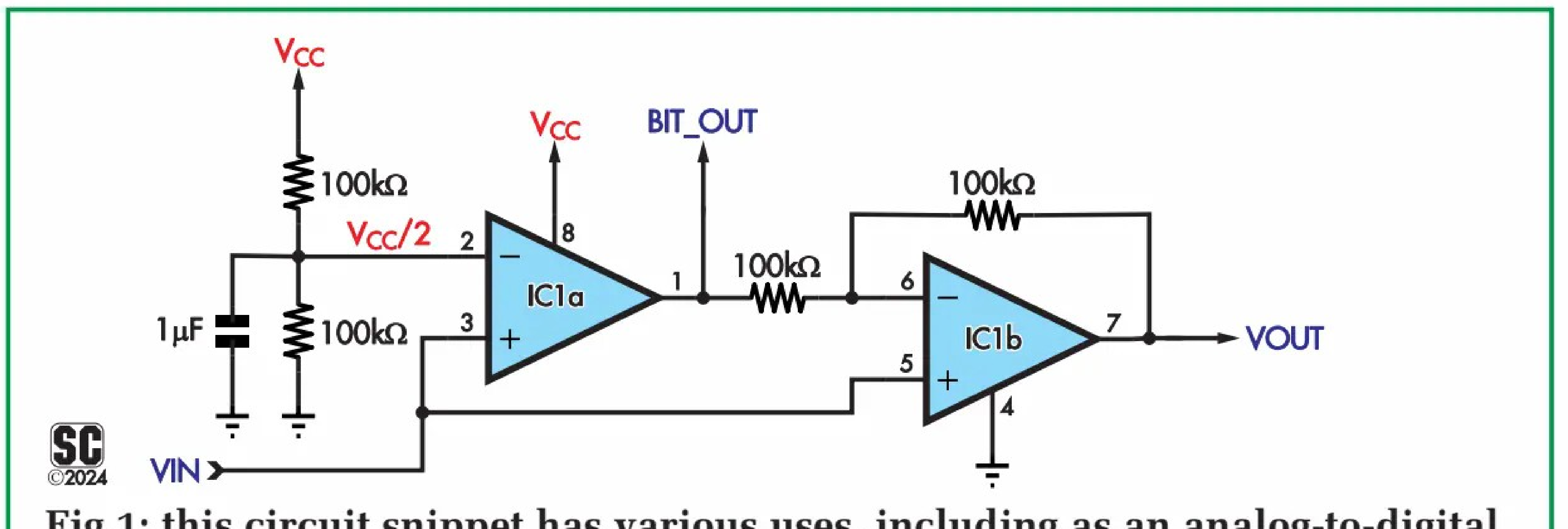
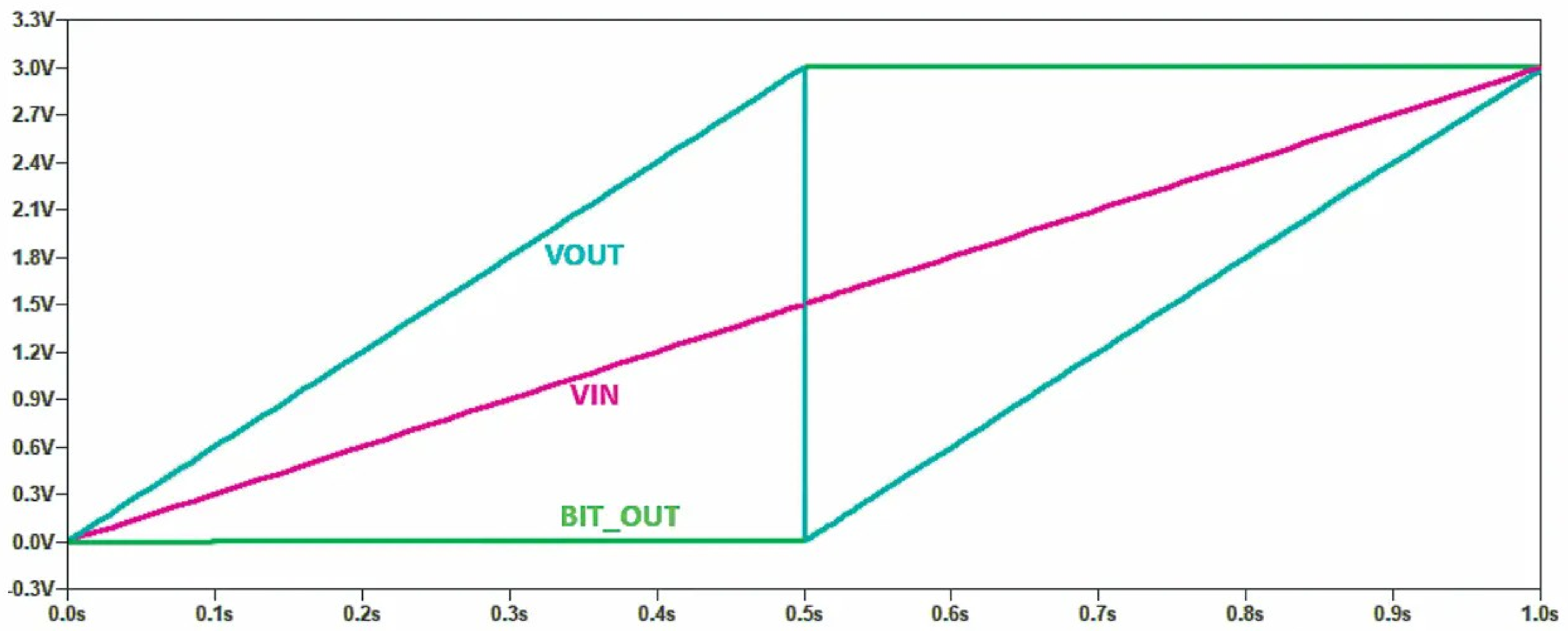
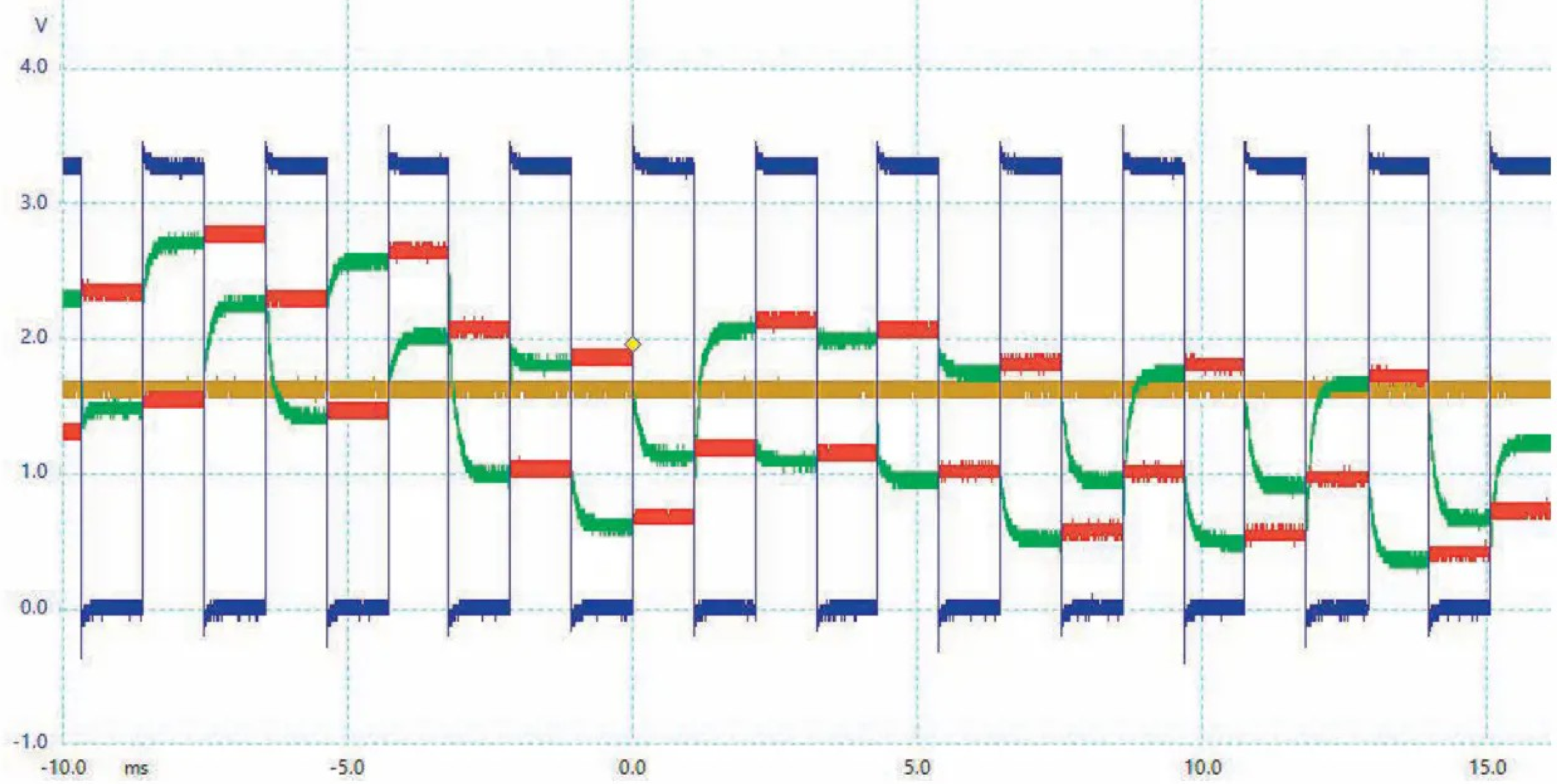


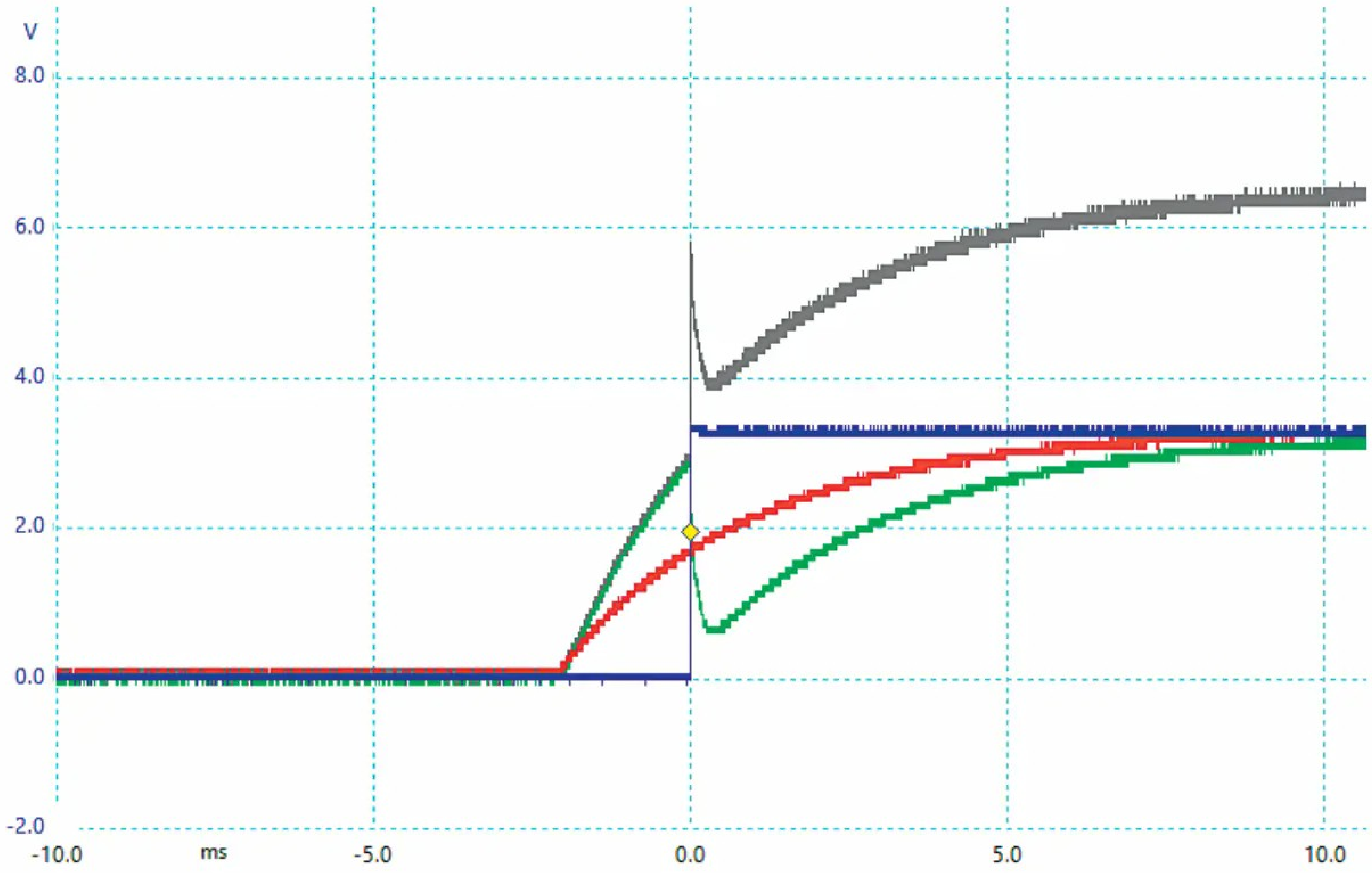
Fig.1: this circuit snippet has various uses, including as an analog-to-digital converter, but we are using it as a noise source. By sampling and holding the output and feeding it back to the input, repeated cycles amplify the noise to a measurable level.



Scope 1: these traces are from a simulation of the Fig.1 circuit, with V_{IN} being the input and V_{OUT} and BIT_OUT being the outputs. If you add the green trace to the cyan trace, the result is double the pink trace.



Scope 2: the voltages around IC2 and IC3. The blue trace is one of the phase outputs from IC1, while the yellow trace is the mid-rail reference. The red trace is V_{IN} and the green trace is V_{OUT} downstream of the 1kΩ resistor. Note the settling time and that for each phase, the red trace follows the previous phase's green trace as the capacitors alternate.



Scope 3: this is much the same as Scope 1, except it is measured on the actual hardware. V_{IN} (red) ramps up as current is applied to one of the 100nF capacitors. The transition on BIT_OUT (blue trace) is clear. The grey trace shows the sum of BIT_OUT and V_{OUT} (green trace), which is double V_{IN} apart from the brief glitch at the transition.

supply rails, the circuit avoids saturating, which would cause noise information to be 'lost'. The extra information is available at BIT_OUT.

Another name we have seen for this circuit is "modular entropy multiplier". Consider a division operation, with the dividend being V_{IN} × 2 and the divisor being V_{CC}. The outputs (BIT_OUT and V_{OUT}, respectively) are the quotient and modulus (or remainder) of the operation.

Another way of viewing the circuit is as a one-bit analog-to-digital converter. The circuit can iterate over multiple bits by taking the output voltage and feeding it back to the input. To do that, we need a sample-and-hold circuit to allow the intermediate states to stabilise and not immediately feed back.

At each stage, the value of the BIT_OUT line state would be noted, then the voltage on V_{OUT} would be fed back to V_{IN}.

In this case, it turns out that the BIT_OUT values will form a binary value representing the initial voltage. Table 1 shows the progression with a starting voltage of 0.333 (for V_{CC} = 1V). The binary value formed from the BIT_OUT column is 01010101, or 85 in decimal, which is one-third of 256, as expected.

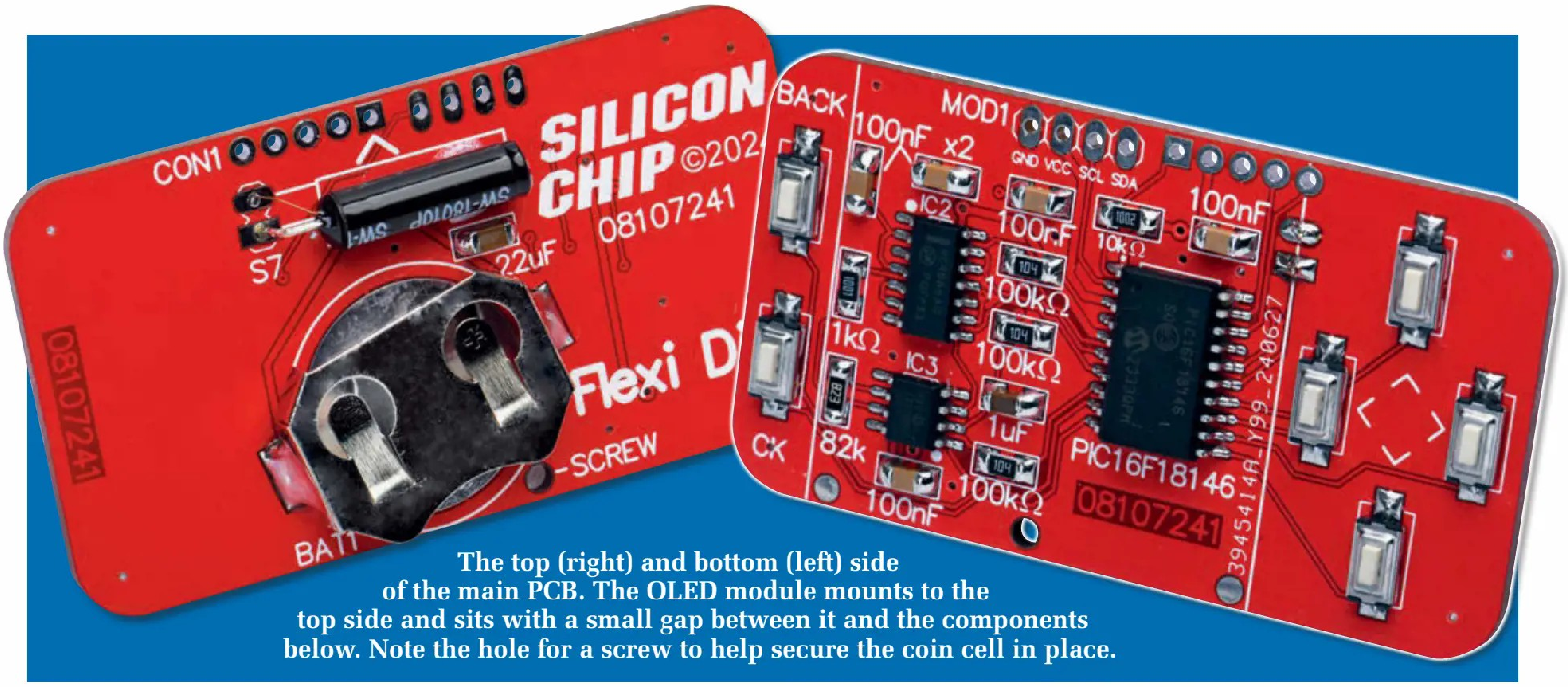
Note that the V_{IN} values do not return exactly to the 0.333 starting value but quickly diverge from it. That is what makes this circuit useful as a random source.

For example, take a 100nF capacitor at 3V, for which the formula Q = CV gives a charge of 3×10^{-7} coulombs or around 1.8×10^{12} electrons. That many electrons can be represented by a binary number with 41 bits. If we run the noise multiplier for more than 41 cycles, we are apparently counting fractions of electrons.

Those familiar with electrons will know that they do not divide easily! What we are measuring at this stage (and probably for many stages before) is just the noise present in the system. That is the essence of the noise multiplier's operation.

Firmware

The program on the microcontroller is responsible for driving the display in response to user input. The main task is to emulate a random event, such as rolling dice or picking a playing card at random.



The top (right) and bottom (left) side of the main PCB. The OLED module mounts to the top side and sits with a small gap between it and the components below. Note the hole for a screw to help secure the coin cell in place.

The firmware requests multiple bits from the noise multiplier by toggling PH1 and PH2 and reading BIT_OUT a few times (more on this later). It performs an XOR operation on those bits. We substantially reduce the correlation between successive bits by combining multiple bits to output one bit.

We need to request multiple bits to represent an event with more than two outcomes. In practice, every roll uses 24 bits from the noise generator. A 24-bit number is large enough that any rounding that might cause one number to appear more often than another is minimal. The result is converted to a coin flip, dice pips, card selection or numerical display and then shown on the OLED screen.

Each ‘roll’ can be configured to show one or two results, and they can be any of the alternatives; you could request a coin and a playing card, for example. Two dice would be a common option.

Note that the choice is done ‘with replacement’. For playing cards, it is equivalent to picking a card from a deck and then returning that card before choosing the second card. Thus, the same card can be selected twice in the same draw.

Another way to consider this is drawing single cards from two separate decks.

When starting up, the analog voltage on pin 19 (which is not connected to anything) is converted to a value from 0 to 1023. The noise multiplier is run for that many cycles plus another 40, which ensures it is not in a state that can not be predicted by the initial conditions.

When a roll is requested, bits are taken from the noise amplifier while the button is held down, further randomising the outcome. An animation is played with random results from the LFSR before the final roll is displayed using results from the noise multiplier.

We use the LFSR for the animation since the noise multiplier takes some time to generate a result. It’s also possible to use the LFSR as the main random data source for rolls.

When the results are displayed, the microcontroller starts a timer. When the timer expires, all peripherals are shut down, and the microcontroller enters a low-power sleep mode that it can be left in for extended periods without flattening the cell.

We tested this with our Coin Cell Emulator from the November 2024 issue. It registered 0.0µA during sleep, so we are confident that the current consumption when not in use is well below 1µA.

Table 1 – analog-to-digital conversion one bit at a time

V_{IN}	BIT_OUT	V_{OUT}
0.333	0	0.666
0.666	1	0.332
0.332	0	0.664
0.664	1	0.328
0.328	0	0.656
0.656	1	0.312
0.312	0	0.624
0.624	1	0.248

When it is not sleeping, like many such projects, the OLED is the main current draw; how much it draws depends on the brightness setting. We saw up to 10mA total current with the default brightness settings, so it pays to keep the OLED brightness as low as possible.

The rest of the circuitry uses about 1.5mA when it is not sleeping, jumping to 2.5mA while a roll is occurring or SETTINGS is active, since the processor has more to do.

The remainder of the firmware is responsible for configuring the FlexiDice, including choosing what combinations of rolls are available. We’ll delve into these once construction is complete.

Much of the microcontroller’s flash memory (which also holds the program instructions) is used to store the graphics and fonts used to create the various displays.

Circuit details

Fig.2 shows the final FlexiDice circuit. IC2 and IC3 form the noise multiplier, each with a 100nF capacitor bypassing their supplies. IC3 is a dual low-power rail-to-rail op amp configured nearly the same circuit as in Fig.1.

The main exceptions are that the feedback resistor is only 82kΩ and that there is a 1kΩ resistor on V_{OUT} to limit peak currents from the op amp.

The V_{IN} and V_{OUT} voltages connect to quad analog switch IC2, which is arranged to allow either of two 100nF capacitors to be connected to V_{IN} and V_{OUT}. This is the sample-and-hold buffer mentioned before in practice. The PH1 and PH2 lines from microcontroller IC1 control it.

If both PH1 & PH2 are low, the capacitors are disconnected. When PH1 is high and PH2 low, one capacitor is connected to V_{IN} and the other to V_{OUT}. The connections are reversed if PH1 is low and PH2 is high. The situation with both PH1 and PH2 high is avoided.

Alternating PH1 and PH2 allows us to step bits out of the noise multiplier circuit, which you can see in Scope 2. Note the settling time (about 1ms) needed to ensure the capacitor fully charges to the V_{OUT} value.

Even with a rail-to-rail op amp, component tolerances and op amp input offsets could conspire to saturate V_{OUT} to one of the power rails, which would result in the same data being continually delivered. The feedback resistor value has been reduced from 100kΩ to 82kΩ in order to exclude the possibility of the multiplier getting stuck in this state.

This value means that V_{OUT} is limited to between about 10% and 90% of V_{CC}, which makes it more likely for BIT_OUT to change states on each cycle. Thus, the output is not entirely random. Using the terms we mentioned earlier, the outcome is fair but slightly predictable. We handle this by requesting extra random bits in the microcontroller firmware.

Microcontroller

IC1 is a PIC16F18146 8-bit microcontroller; it also has a 100nF supply bypass capacitor. A 22µF bulk bypass capacitor helps reduce the peak current loads on 3V coin cell BAT1.

IC1's pin 4 $\overline{\text{MCLR}}$ input is pulled up to 3V by a 10kΩ resistor; this, the power pins (1 and 20) and programming pins (18 and 19) are taken to ICSP connector CON1. This can be used to program the microcontroller; we also used it for debugging during development.

Coin Cell Precautions

The FlexiDice requires a coin cell; even though we have added protections such as the locking screw, care should be taken so that children are not left unattended with it.

The microcontroller drives the PH1 and PH2 lines from pins 6 and 7, ensuring that both are never high simultaneously. When it switches them, they are both briefly set low to ensure that the circuitry around IC2 is not closed in a loop.

Seven of the micro's I/O pins (9, 10, 14, 13, 12, 11 and 15) are configured as inputs with pullups, and these connect to switches S1-S7.

S1-S6 are tactile switches, while S7 is a vibration switch that can be triggered by shaking or bumping the FlexiDice. The other ends of all the switches connect to ground so that their closure changes the state of the connected I/O pin to low.

MOD1 is an I²C OLED display module powered by IC1's pin 3 (RA4). This allows the display to be completely powered down for minimum power consumption when not needed. Pins 2 and 16 of IC1 provide the I²C data interface for updating the display.

Similarly, the noise multiplier is powered from IC1's pin 5 and can be shut down as needed. We have con-

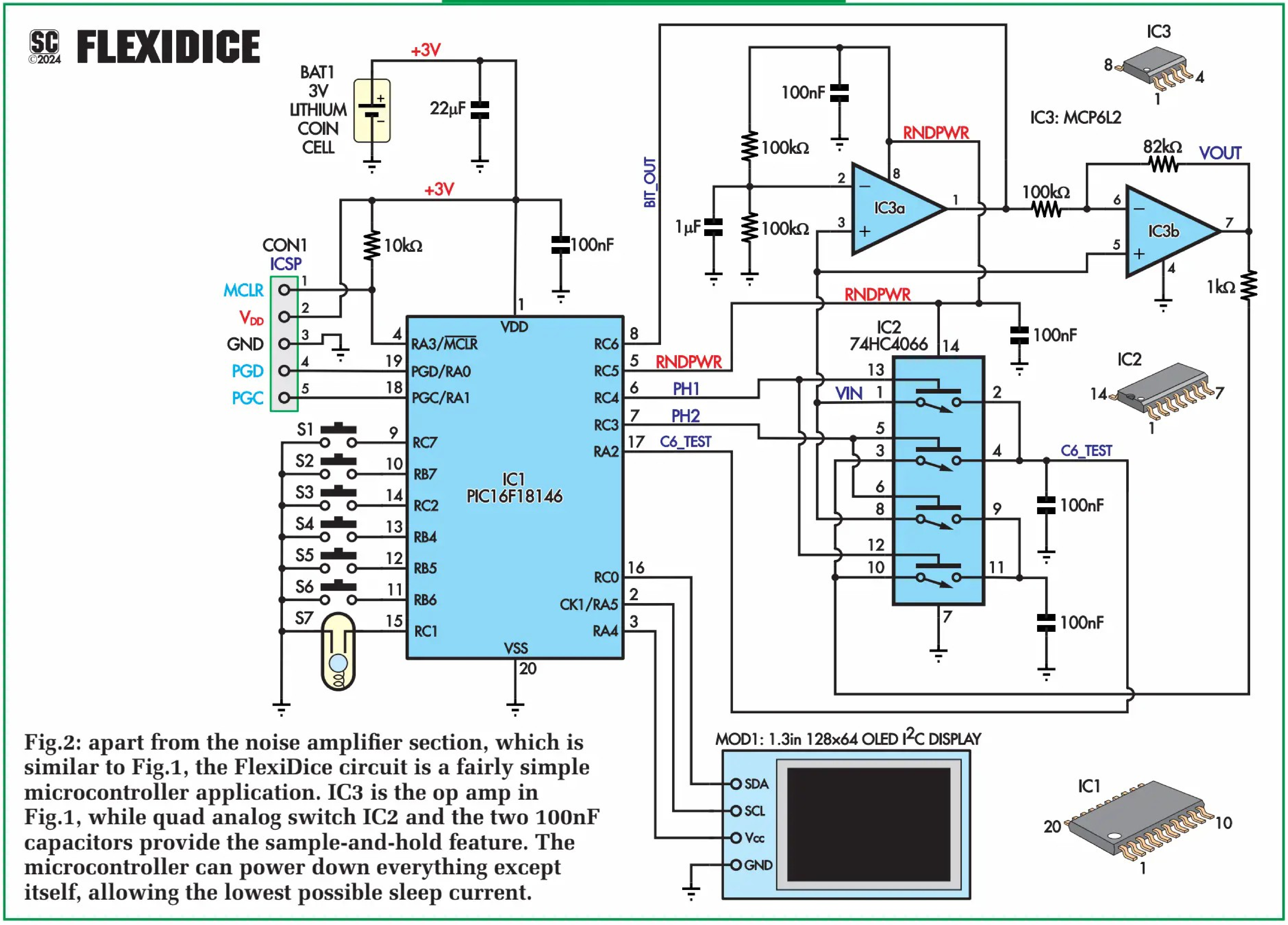

SC ©2024 FLEXIDICE
BAT1 3V LITHIUM COIN CELL
+3V
22µF
+3V
10kΩ
100nF
CON1 ICSP
MCLR
VDD
GND
PGD
PGC
S1
S2
S3
S4
S5
S6
S7
VDD
RA3/MCLR
PGD/RA0
PGC/RA1
RC7
RB7
RC2
RB4
RB5
RB6
RC1
VSS
RC6
RC5
RC4
RC3
RA2
RC0
CK1/RA5
RA4
IC1 PIC16F18146
BIT_OUT
RNDPWR
PH1
PH2
C6_TEST
100nF
100kΩ
1µF
100kΩ
IC3a
RNDPWR
100kΩ
82kΩ
VOUT
IC3b
1kΩ
IC3
IC3: MCP6L2
RNDPWR
IC2 74HC4066
100nF
VIN
100nF
100nF
IC2
MOD1: 1.3in 128x64 OLED I²C DISPLAY
SDA
SCL
Vcc
GND
IC1

Fig.2: apart from the noise amplifier section, which is similar to Fig.1, the FlexiDice circuit is a fairly simple microcontroller application. IC3 is the op amp in Fig.1, while quad analog switch IC2 and the two 100nF capacitors provide the sample-and-hold feature. The microcontroller can power down everything except itself, allowing the lowest possible sleep current.

nected one of the noise multiplier capacitors to pin 17, allowing us to monitor the noise multiplier state or inject a voltage if required.

Using a weak pullup current from pin 17, we created the plot shown in Scope 3. This is similar to Scope 1 but measured on real hardware instead of a simulation.

All ICs are rated for operation down to 2V or lower, but from experience, we have found that the OLED display modules will falter around 2.4V; this is what sets our lower operating limit. A lithium cell reaching that voltage under a light load has exhausted almost all its stored energy.

Parts List – FlexiDice

1 double-sided main PCB coded 08107241, 34 × 62mm
1 double-sided panel PCB coded 08107242, 34 × 62mm
1 SMD 2025/2032 coin cell holder (BAT1)
1 5-way right-angled header strip (CON1, **optional**, for ICSP)
1 1.3in 128×64 I²C OLED module (MOD1)
1 4-way pin header (for MOD1, may be included)
6 SMD 2-pin tactile switches (S1-S6)
1 SW18010 vibration-triggered switch or similar (S7)
1 M2 × 6mm Nylon panhead machine screw
2 M2 Nylon hex nuts
1 2 × 2cm piece of double-sided foam-core tape
2 1cm piece of wire (eg axial lead offcut or pin header) to secure MOD1

Semiconductors
1 PIC16F18146-I/SO microcontroller programmed with 0810724A.HEX, SOIC-20 (IC1)
1 74HC4066 quad analog mux IC, SOIC-14 (IC2)
1 MCP6L2 dual low power rail-to-rail op amp, SOIC-8 (IC3)

Capacitors *(all SMD M3216/1206 X5R/X7R)*
1 22µF 16V
1 1µF 35V
5 100nF 50V

Resistors *(all SMD M3216/1206, 1%, ⅛W)*
3 100kΩ [104/1003]
1 10kΩ [103/1002]
1 82kΩ [823/8202]
1 1kΩ [102/1001]

Construction

The FlexiDice is built on a double-sided PCB coded 08107241 that measures 34 × 62mm.

It includes surface-mounting components, so you will need the standard SMT gear. A fine-tipped soldering iron, flux paste and tweezers are recommended. A magnifier, some solder-wicking braid and fume extraction will also help.

Start with the three ICs. They are all different sizes, so it should be easy to tell them apart, although you will have to take care with their orientations. Note the location of the pin 1 dot in each case and check it against the PCB silkscreen and Fig.3 overlay diagram.

Start by applying flux to the PCB pads for the ICs and sit each in place. Tack one lead and check that the others are aligned. If not, remelt the solder and nudge them into place. Also make sure that the parts are flat against the PCB.

Solder the remaining leads, cleaning the iron's tip as needed. If you get a bridge between pins, you can remove that by adding more flux and pressing the braid against the bridge with the iron. Carefully drag both away together once the excess solder has been taken up.

Five 100nF capacitors and one 1µF capacitor mount on the top of the PCB. Do not mix them up, as they will not be marked. Solder them using a similar strategy to the ICs: add flux, tack one lead, check and then solder the other lead.

The six resistors will have codes marked on their tops, making them less likely to be mixed up. Solder them to the PCB similarly, according to the silkscreen markings.

Flip the PCB over and solder the 22µF capacitor, followed by cell holder BAT1, as shown in Fig.3. Ensure that the cell holder opening faces towards the edge of the PCB. Add a good amount of solder to help secure it firmly.

Now is a good time to clean flux residue from the PCB. Use a solvent such as isopropyl alcohol, a general flux cleaner or whatever is recommended for your flux. Allow the PCB to dry and inspect it closely for dry joints, bridges and other issues. These will be hard to fix once the OLED screen is fitted to the PCB.

One of the more insidious problems occurs when the solder does not adhere to the pad on the PCB. The lead may appear to have a glossy, well-formed solder bead, but it is not connected to the pad below. That can be caused by the part not being flat against the board. If you find this has happened, add more flux and press down gently on the pin with your soldering iron.

Solder the six tactile switches next, being sure to align them with their

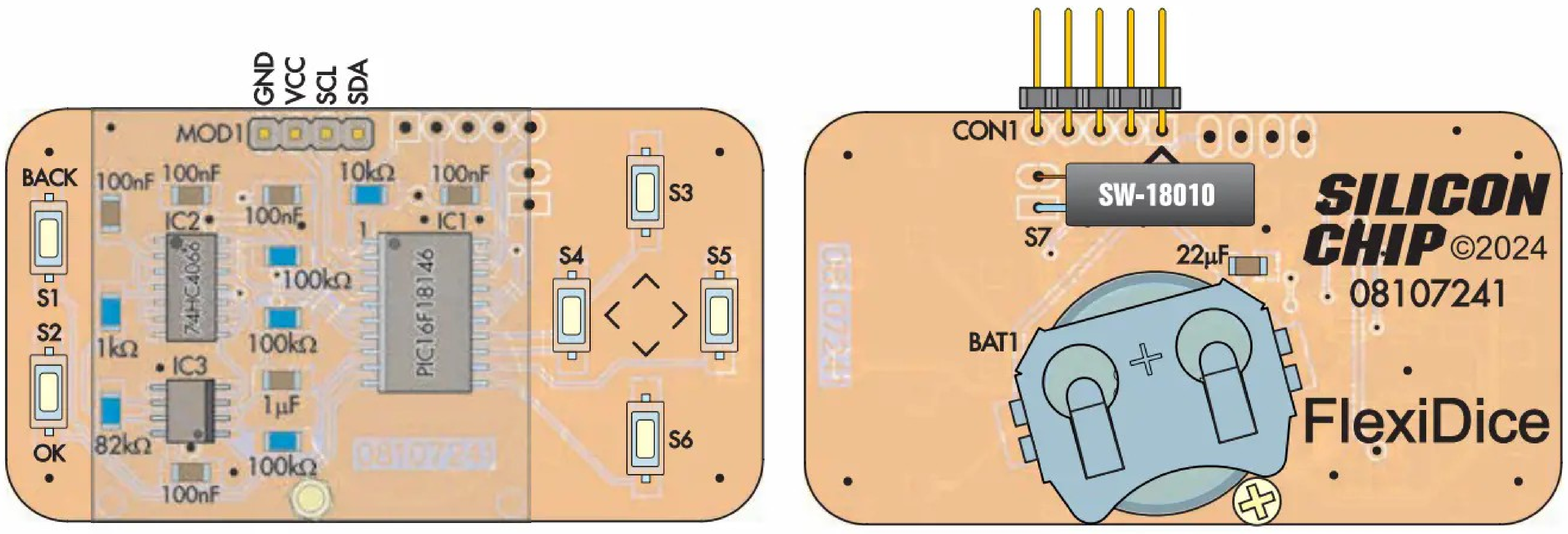
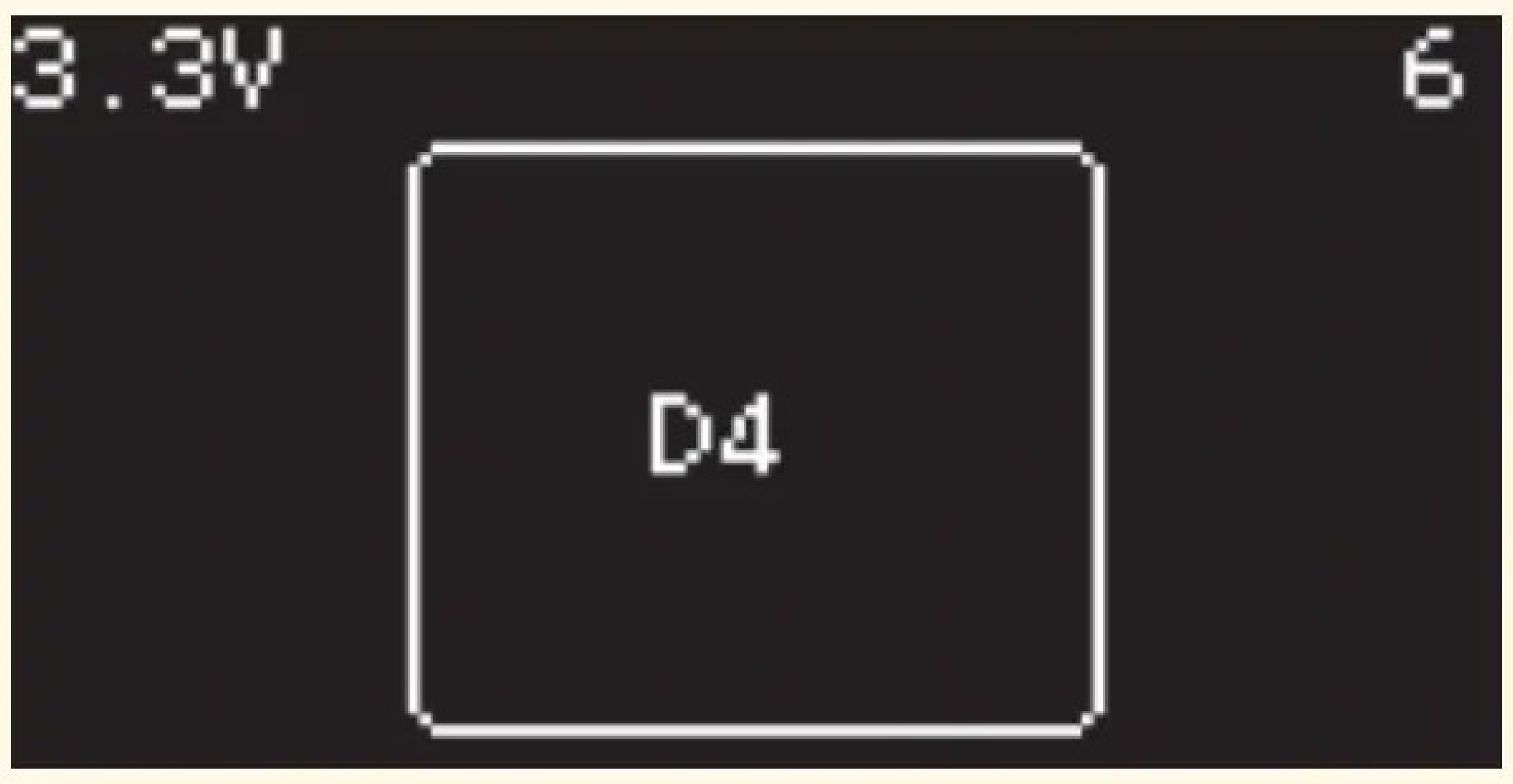
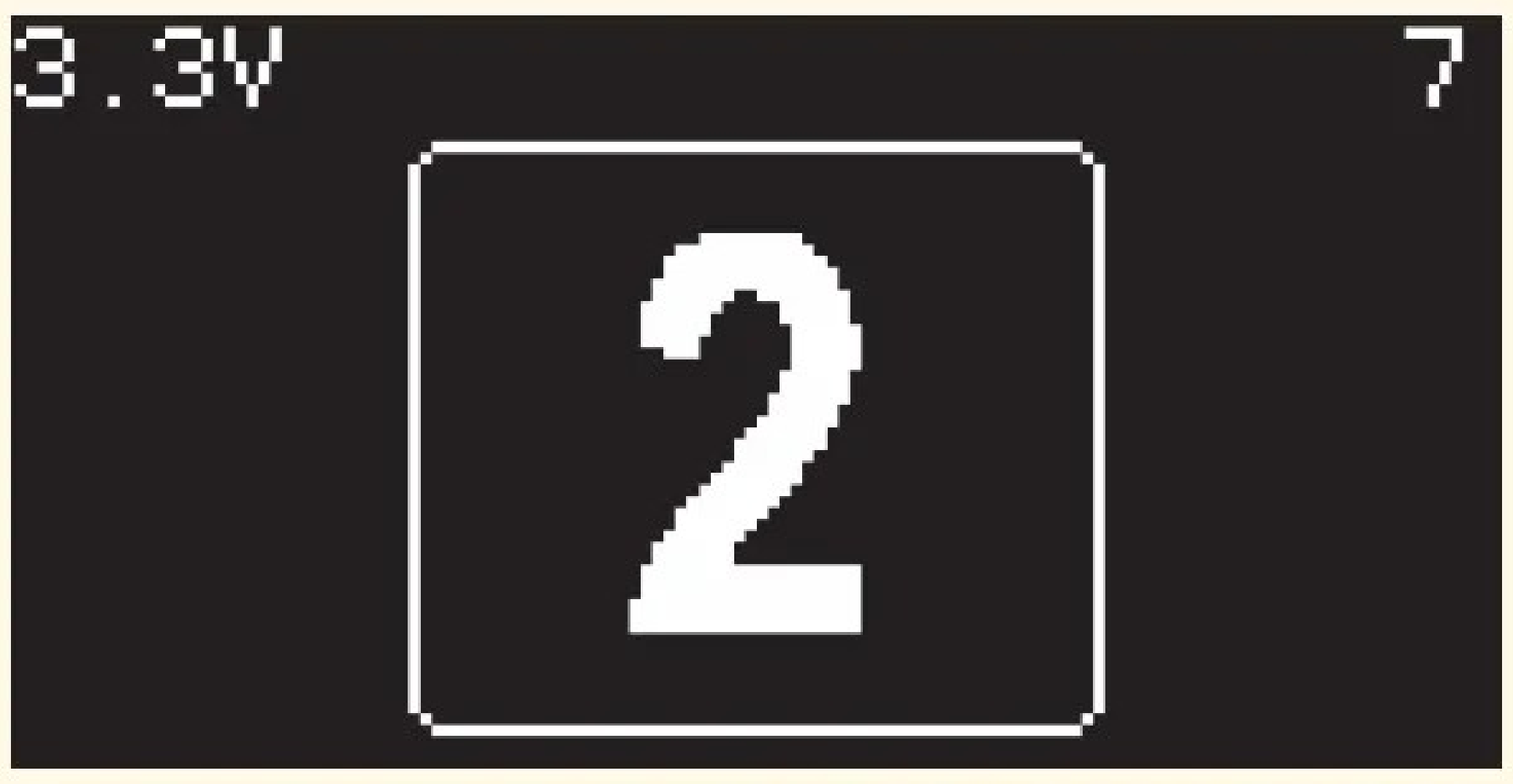


Fig.3: assembly of the FlexiDice is easy with even modest SMD skills. Ensure the ICs are orientated correctly and do not mix up the capacitors. The OLED module sits over the top of the PCB (see the black outline). Once you have tested everything, we recommend carefully glueing the body of the vibration switch to the PCB.


3.3V 6

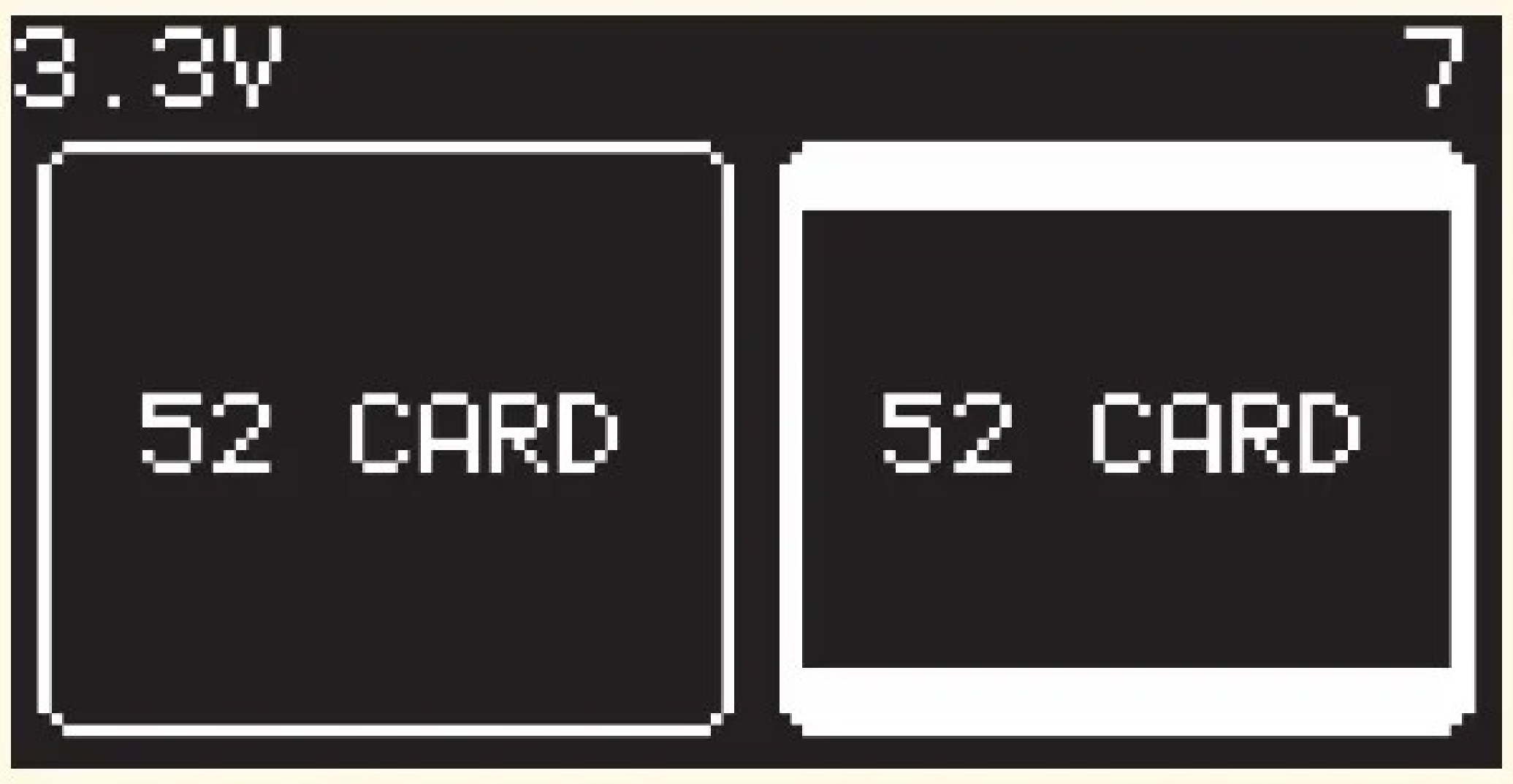
D4

Screen 1: when first powered on, the FlexiDice shows this screen, indicating it is ready to roll a D4 (four-sided die). The coin cell voltage is at upper left, while the sleep countdown timer is at top right.


3.3V 7

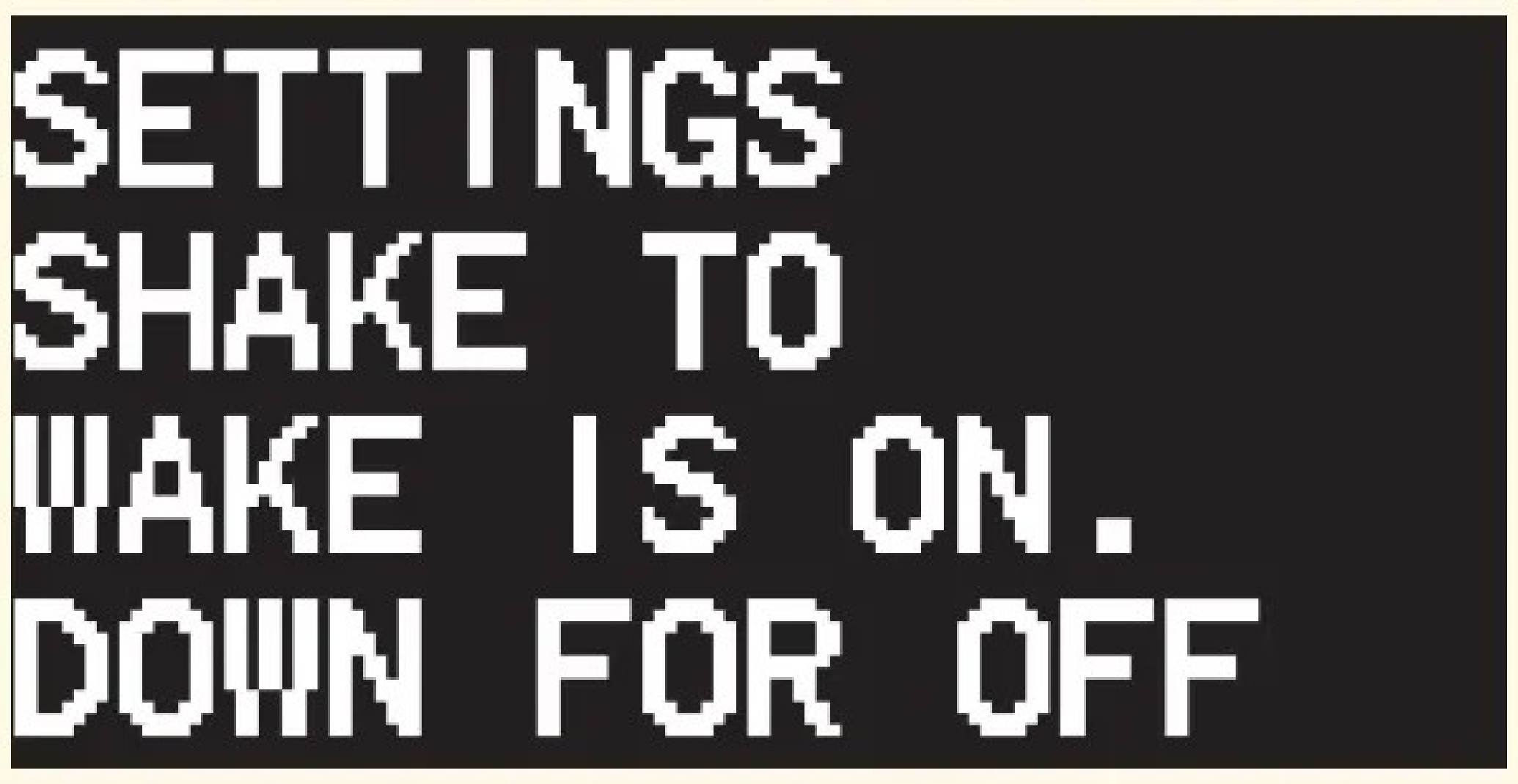
2

Screen 2: pressing the UP button will run a brief random animation and then show the result of the roll. Pressing DOWN will return to Screen 1. Any button press will also reset the countdown timer.

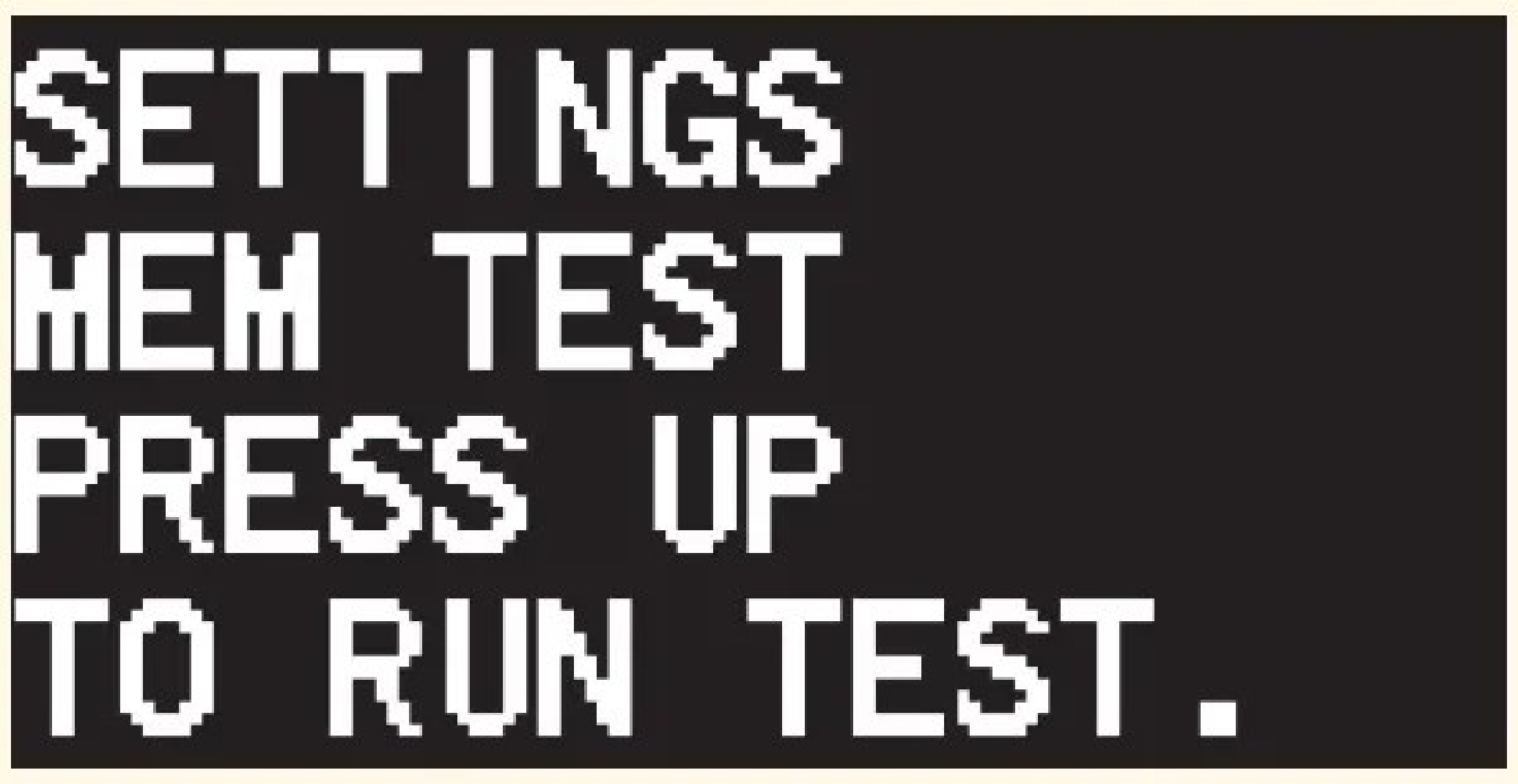

3.3V 7

52 CARD 52 CARD

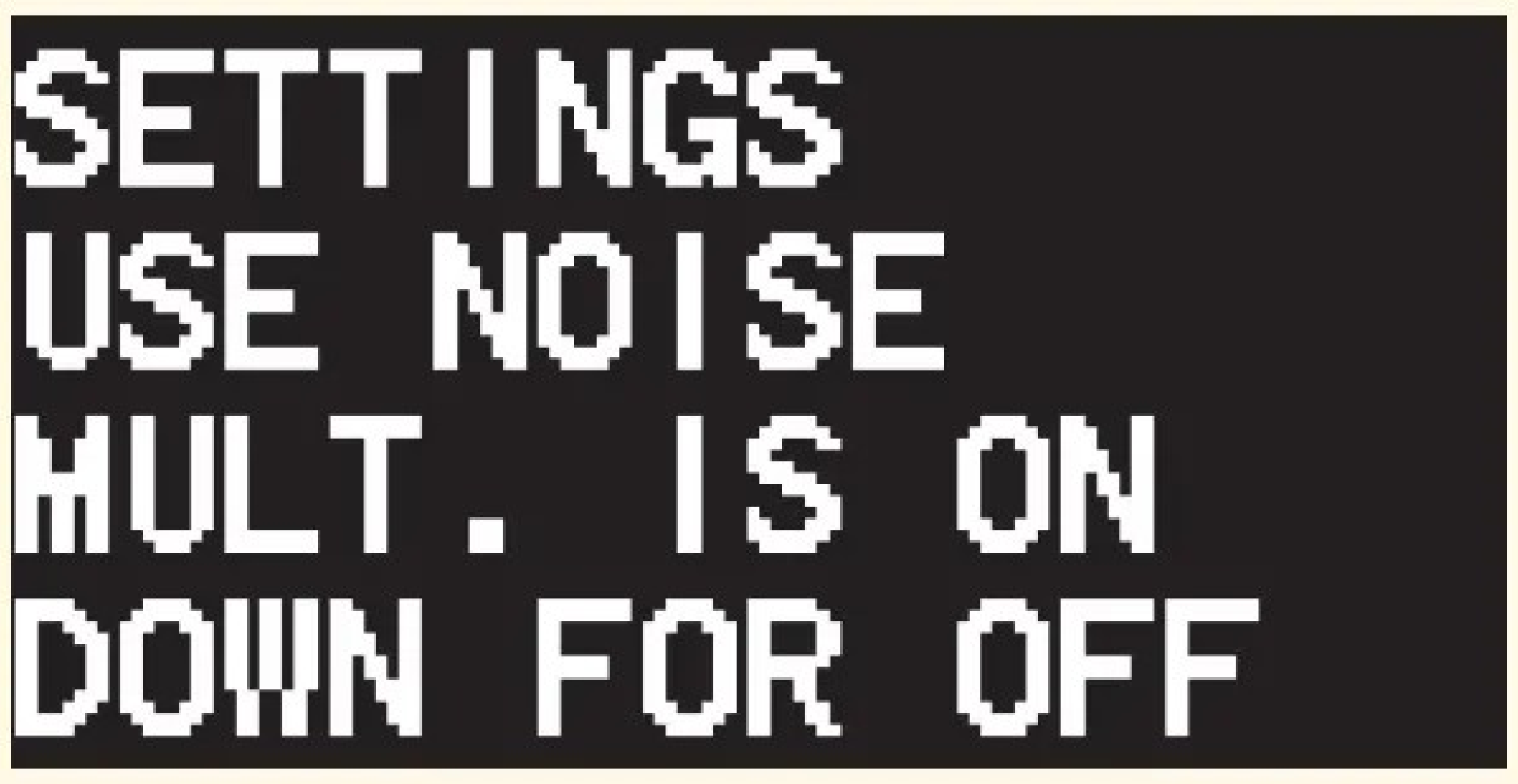
Screen 3: pressing LEFT and RIGHT will cycle between numerous roll options. Shown here is a draw of two playing cards, each from a standard 52-card deck (without Jokers).


SETTINGS
SHAKE TO
WAKE IS ON.
DOWN FOR OFF

Screen 7: in case you find the vibration sensor too sensitive, you can turn off S7's ability to wake the FlexiDice from low-power sleep. By default, 'shake to wake' is on.


SETTINGS
MEM TEST
PRESS UP
TO RUN TEST.

Screen 8: if you wish to test the MEM (modular entropy multiplier), press UP from this screen. Screen 11 shows the testing screen that can be used to check its fairness and correlation.


SETTINGS
USE NOISE
MULT. IS ON
DOWN FOR OFF

Screen 9: if you prefer to use the LFSR (linear feedback shift register) as the random noise source, this screen can be used to turn the hardware noise source off, saving a small amount of power.

silkscreen markings. Any excess flux can be cleaned up with a cotton tip dipped in solvent, which avoids getting solvent into the switch mechanisms (that can cause them to fail).

Solder vibration switch S7 next. Bend the leads 90°, being mindful of the orientation of the leads. Make sure its body is flat against the PCB.

If the 4-way header is not already attached to MOD1, the OLED module, fit it now. Then use a piece of card or thin plastic to temporarily space the module away from the components below it on the PCB. Solder its leads, adjusting if needed to make the display align neatly with the PCB.

Trim the excess lead lengths and remove the card or plastic. Next, solder some short pieces of wire (such as lead offcuts or single header pins) from the PCB to the two pads in the bottom corners of the display, adding some mounting rigidity.

Programming IC1

If you have bought the PIC or a kit from the Silicon Chip Shop, IC1 will be programmed already and you can skip to the next section.

Otherwise, use a PICkit 4, PICkit 5 or Snap programmer and the MPLAB IPE (integrated programming environment). The IPE can be downloaded as part of the MPLAB X IDE from:
https://pemag.au/link/abzy

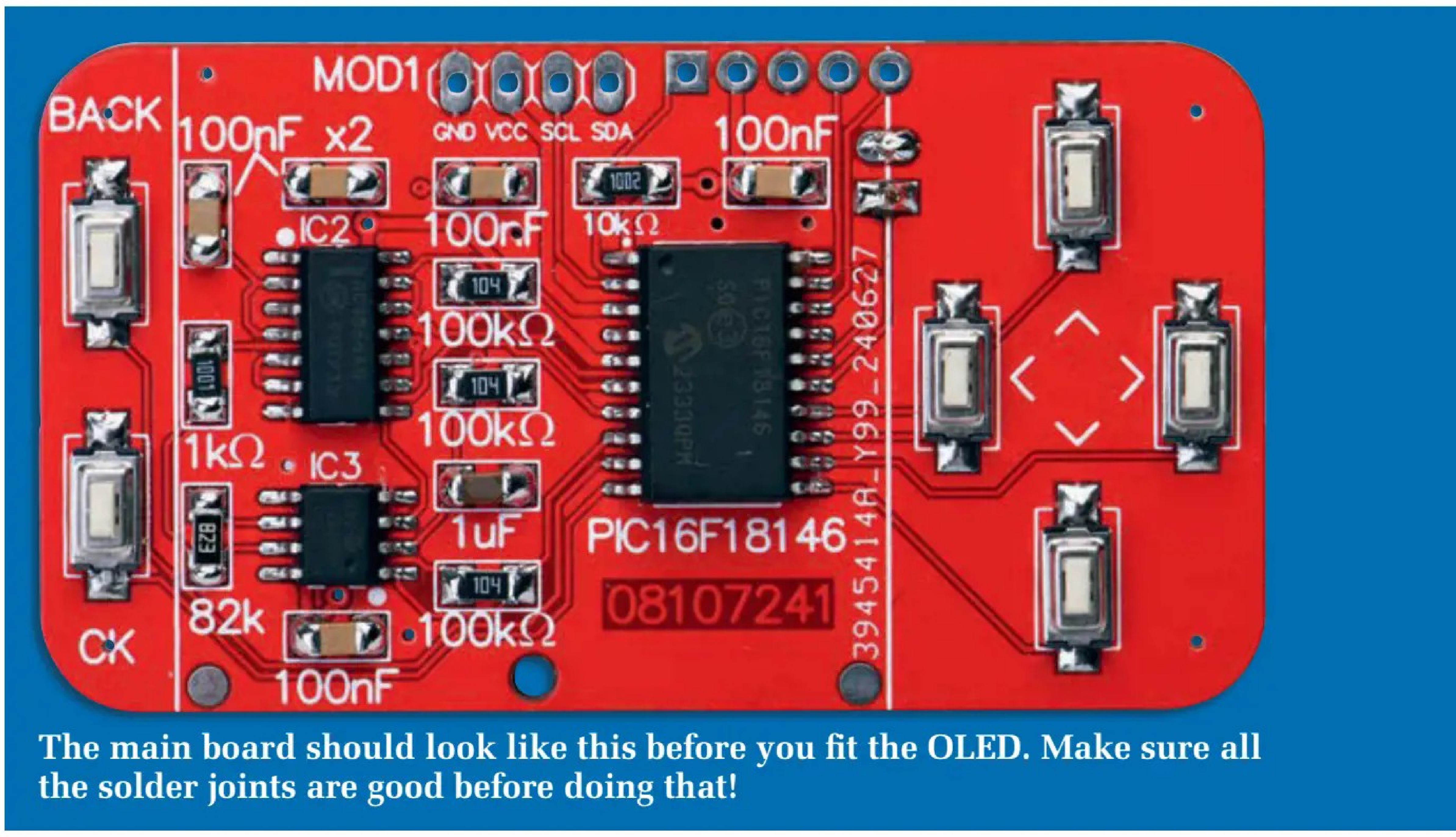
The Snap cannot supply power, so you will need to provide some; fitting a coin cell is the easiest way. Choose the PIC16F18146 as the Part, open the 0810724A.HEX file (available to download from our website) and connect the programmer to CON1, aligning the pins marked with the arrow.

If you only plan to use this connection once, you can insert a five-way pin header into the first five pins of the programmer's header and hold the FlexiDice PCB against the pins to ensure good contact.

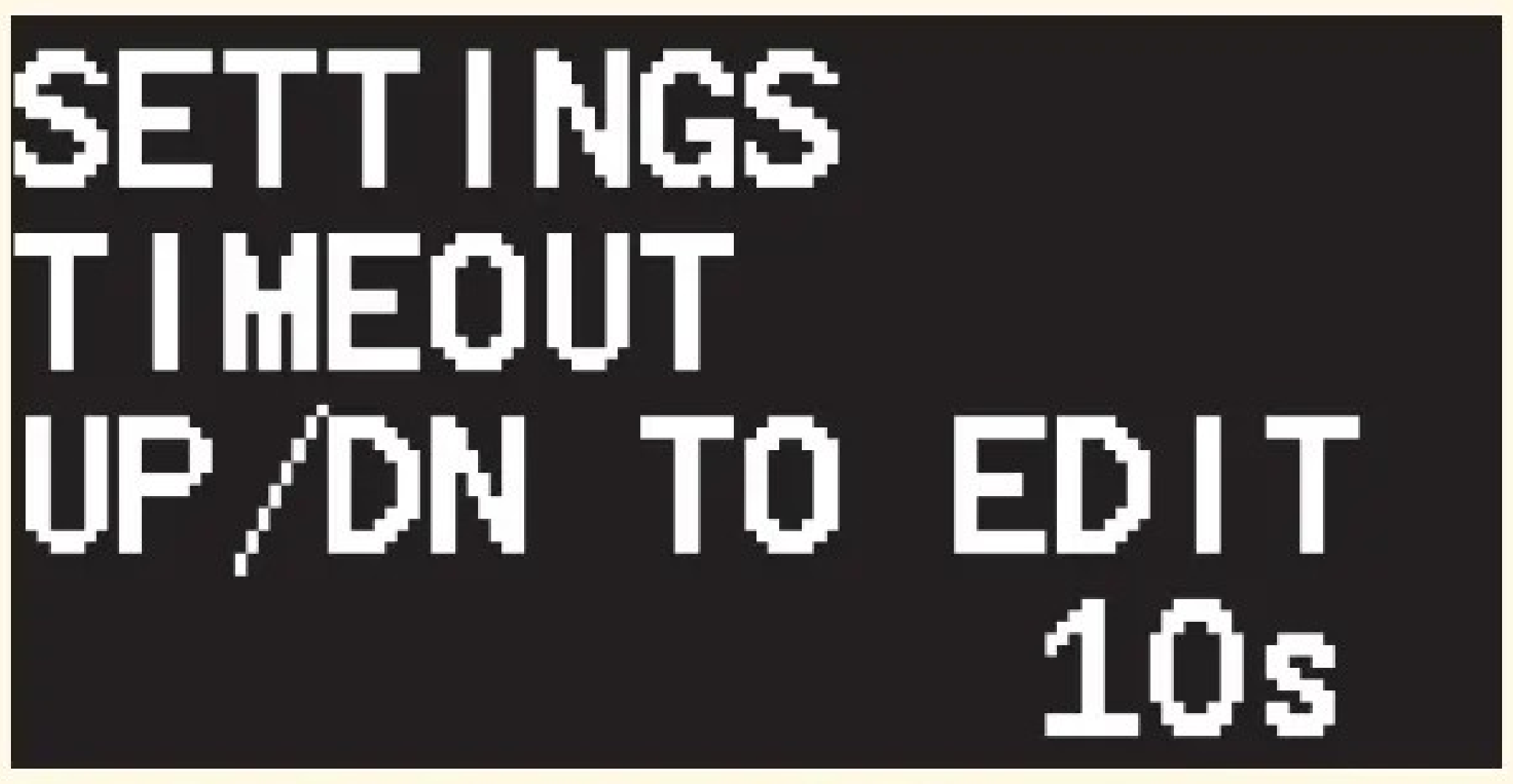
Press the Program button and check that the programming completes and is verified successfully. The OLED should also light up (see Screen 1), indicating that the program is working.

Using it

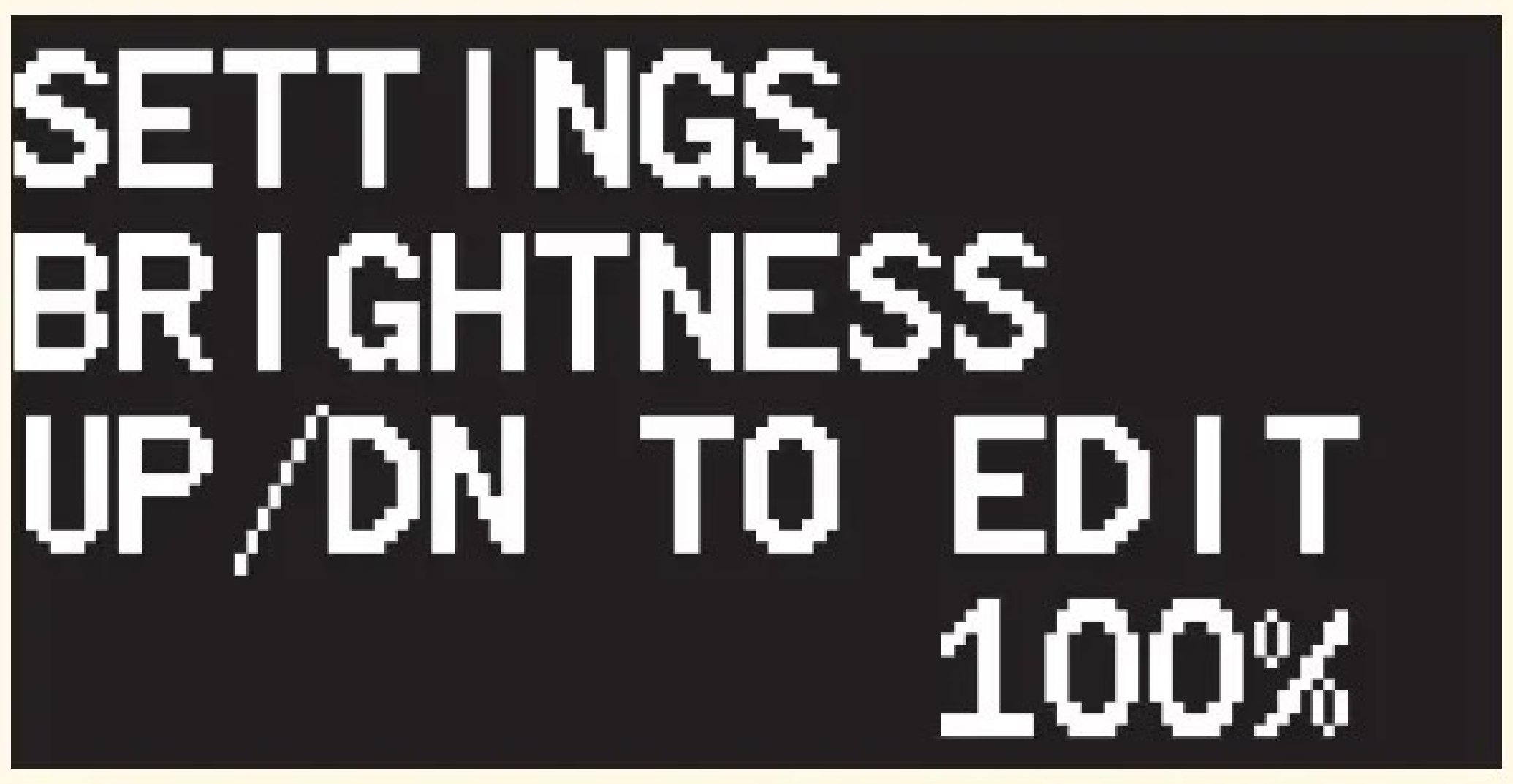
Fit a coin cell if you have not already done so. Check that the polarity is correct; there should be a small + sign on the top part of the cell holder. The default program allows the FlexiDice



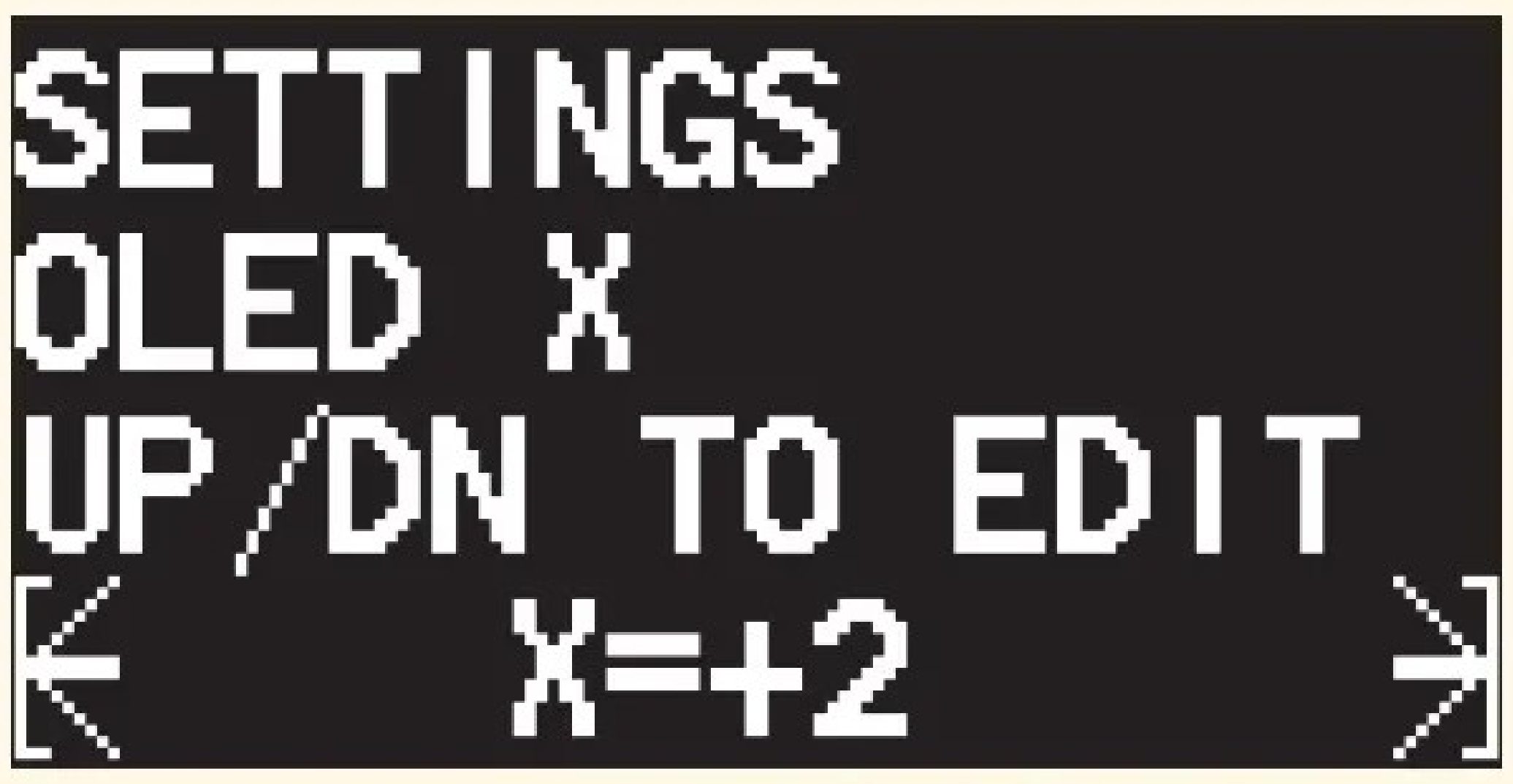
The main board should look like this before you fit the OLED. Make sure all the solder joints are good before doing that!


```
SETTINGS
TIMEOUT
UP/DN TO EDIT
          10s
```

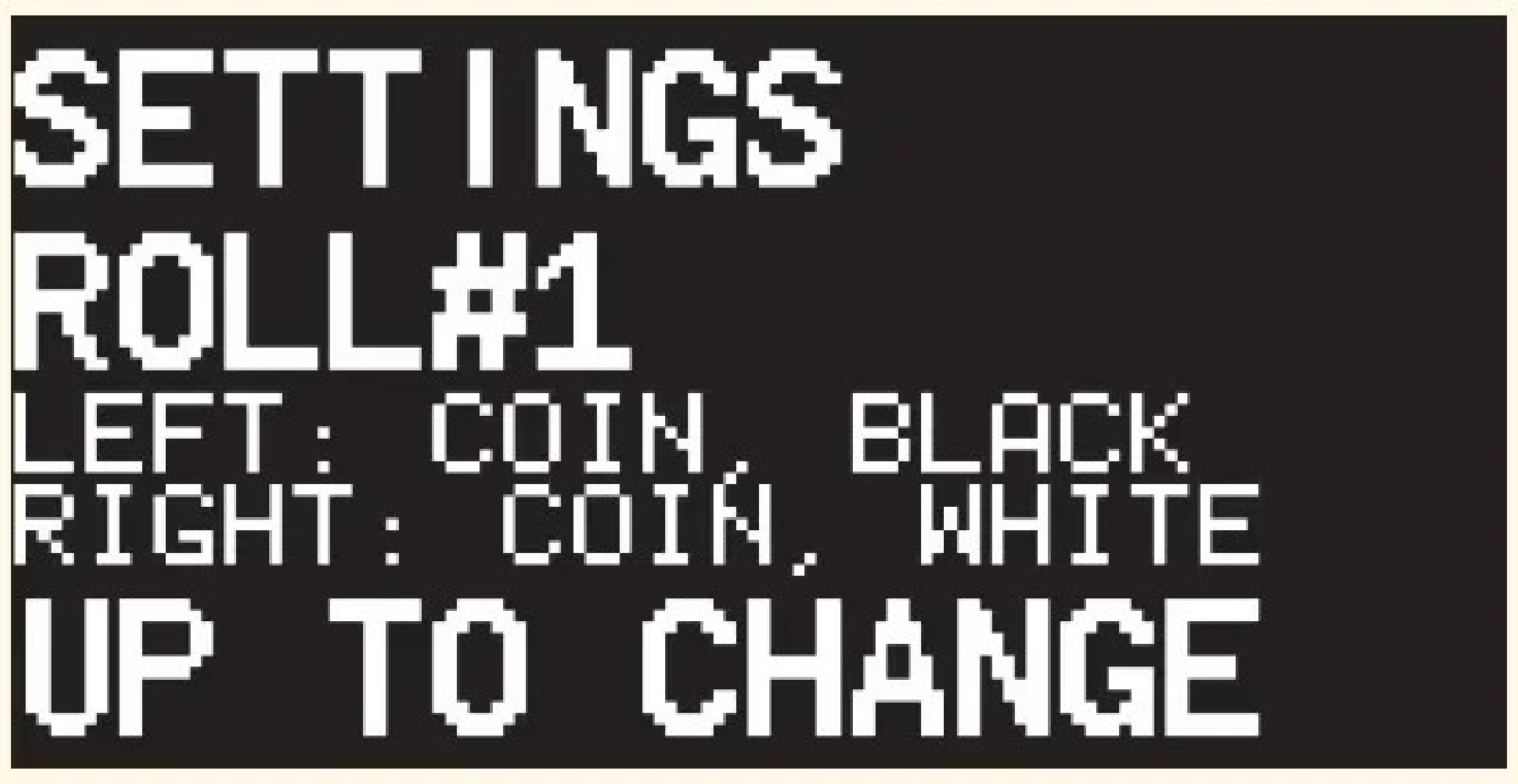
Screen 4: Pressing BACK and OK together will enter SETTINGS. This screen sets the display timeout, which can be changed in five-second steps with the UP and DOWN buttons.


```
SETTINGS
BRIGHTNESS
UP/DN TO EDIT
         100%
```

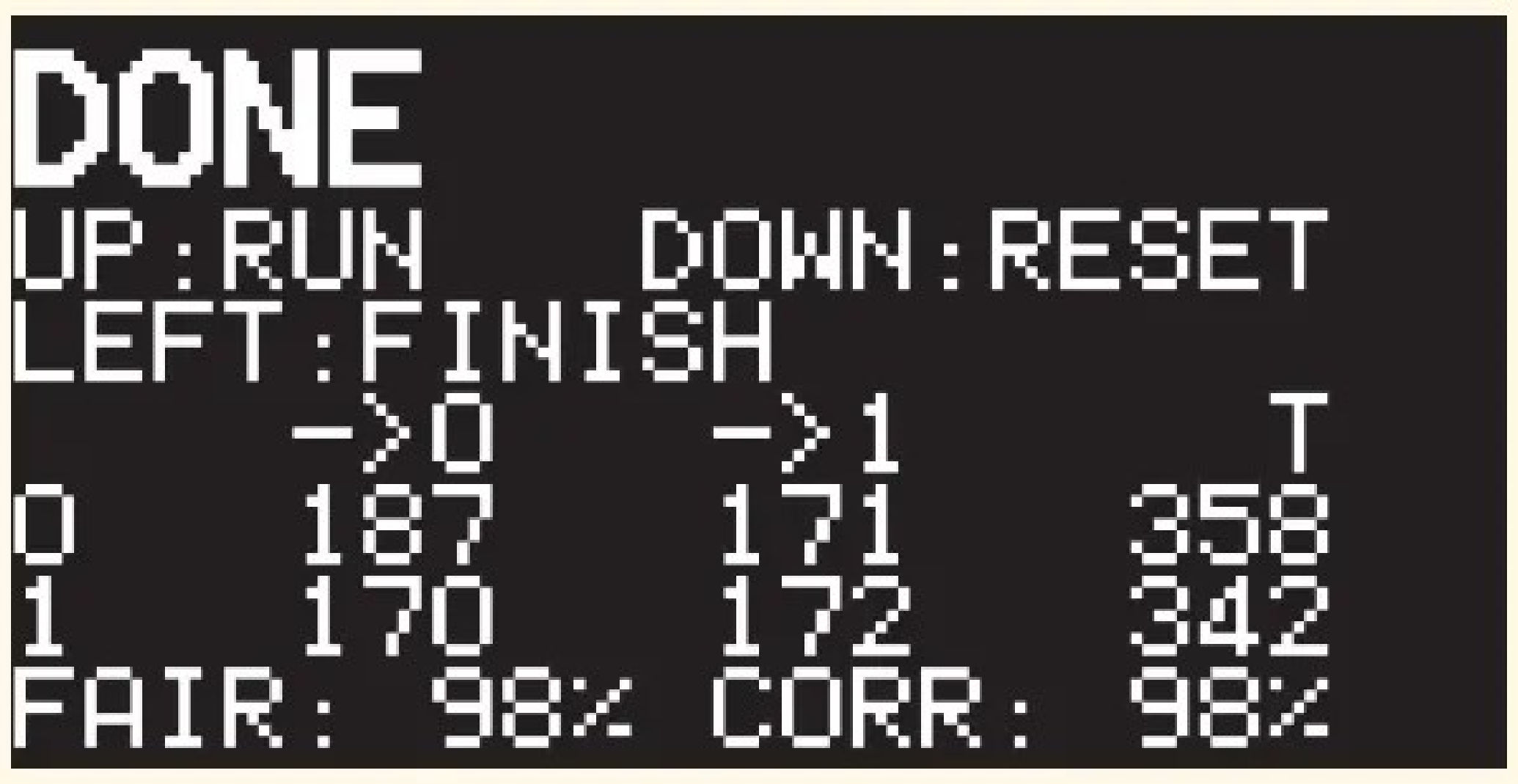
Screen 5: pressing RIGHT cycles to the next setting screen, which changes the display OLED module's brightness. The display will dim slightly during the last two seconds before sleep.


```
SETTINGS
OLED X
UP/DN TO EDIT
     X=+2
```

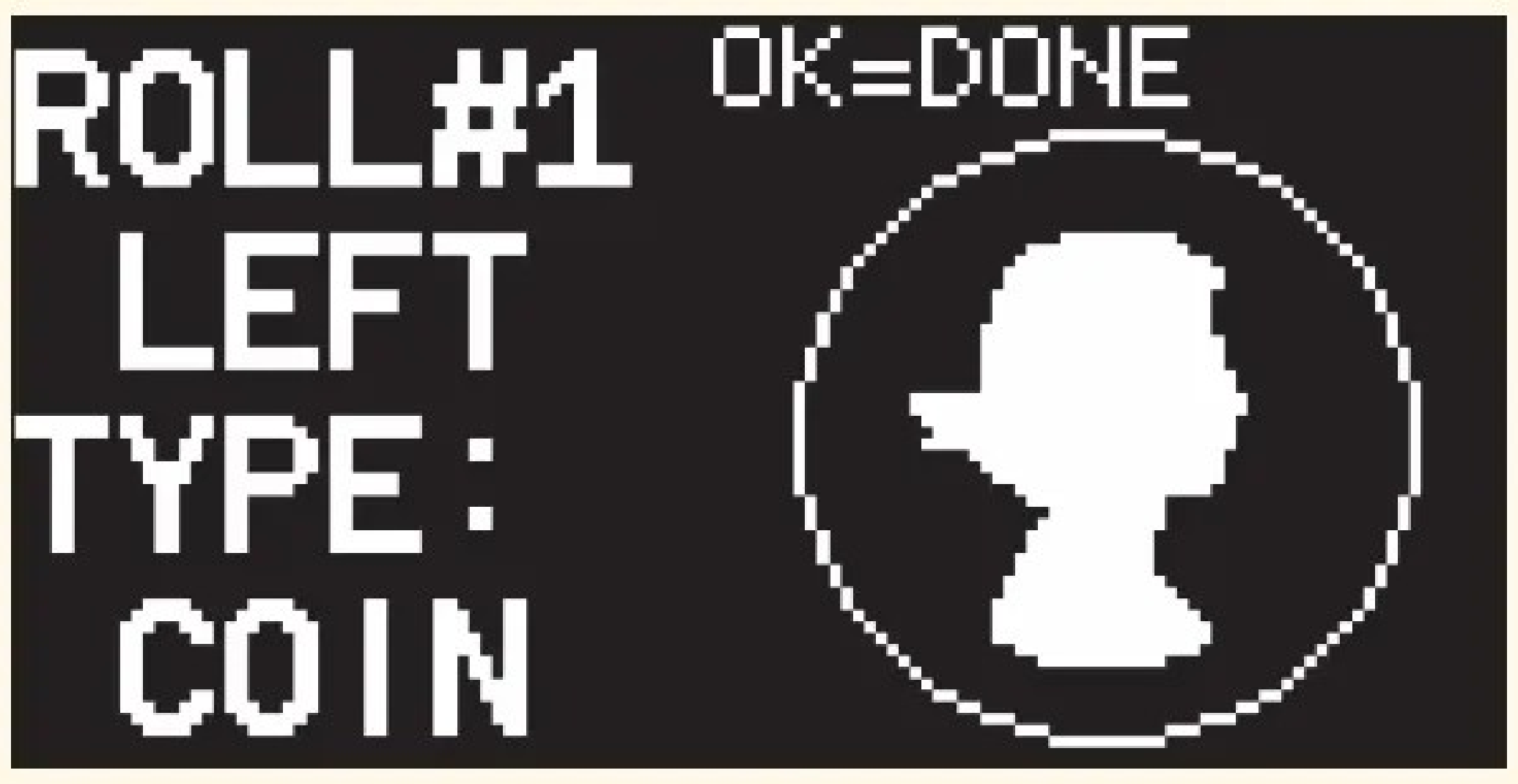
Screen 6: some OLED modules have a horizontal offset, which can be trimmed on this screen with UP and DOWN. Both arrows are showing fully, meaning the display is correctly aligned.


```
SETTINGS
ROLL#1
LEFT: COIN, BLACK
RIGHT: COIN, WHITE
UP TO CHANGE
```

Screen 10: this and nine other screens like it configure your custom rolls. Pressing UP will take you to Screen 12, where you can change the graphics, colour and number of outcomes.


```
DONE
UP:RUN     DOWN:RESET
LEFT:FINISH
    ->0     ->1      T
0   187     171    358
1   170     172    342
FAIR: 98%  CORR:   98%
```

Screen 11: here, UP starts the test, taking 100 single-bit samples from the noise generator. The results are shown at the bottom. If you consistently see low % results, the noise generator may not be working.


```
ROLL#1  OK=DONE
 LEFT
TYPE:
 COIN
```

Screen 12: from each page here, use LEFT and RIGHT to view each option and UP and DOWN to edit it. If the right die is set to NONE, only a single outcome (the left die) will be displayed.

to work immediately, and the OLED should show a sensible display as soon as it is powered on (see Screen 1).

You can initiate a roll by pressing S3 (the UP button) or activating the S7 vibration switch. Screen 2 shows the result of a roll. S4 and S5 (LEFT and RIGHT) cycle between the different roll options. Screen 3 shows one of the other options before a roll is performed.

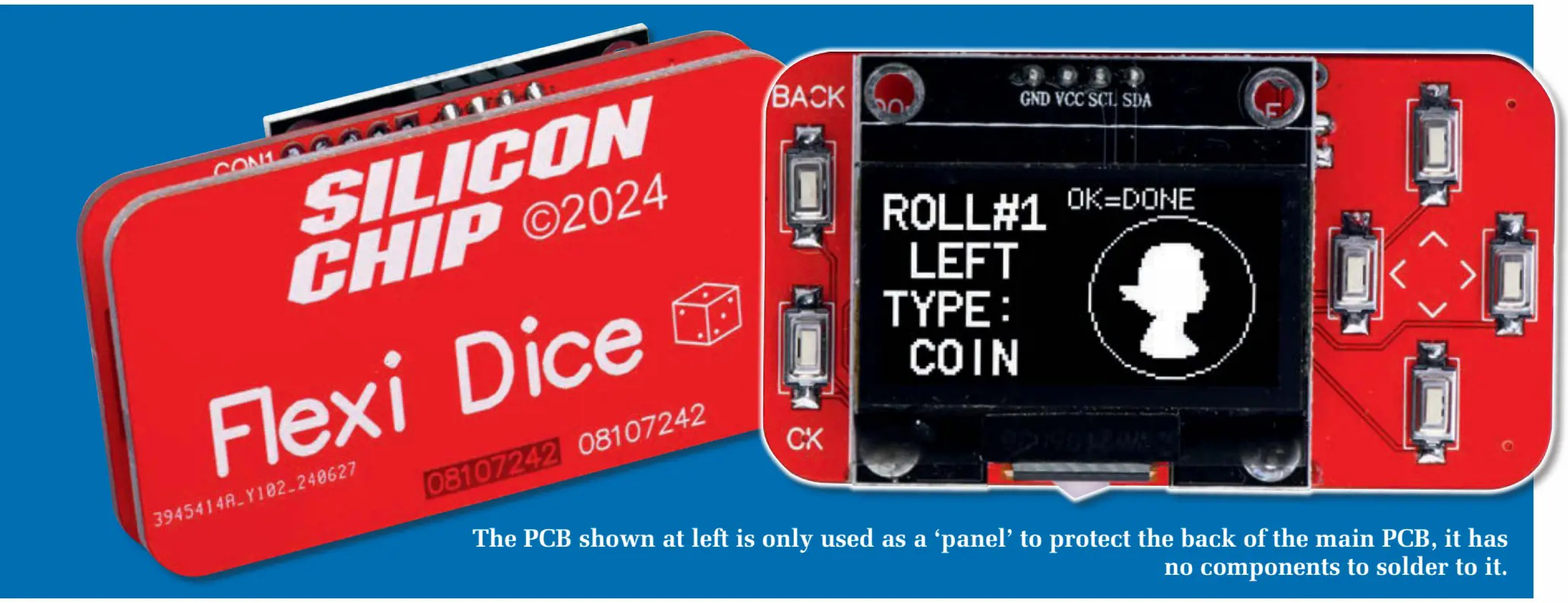
Pressing S6, the DOWN button, will reset the screen; the other two buttons are associated with SETTINGS. If the display times out and the FlexiDice enters sleep mode, pressing any button will wake it up again.

Perform a few rolls and confirm that the results appear random. If you get the same result on every roll (especially if it is 1), the noise multiplier may not be working. There is a utility within SETTINGS to check the noise multiplier's output (see Screen 11).

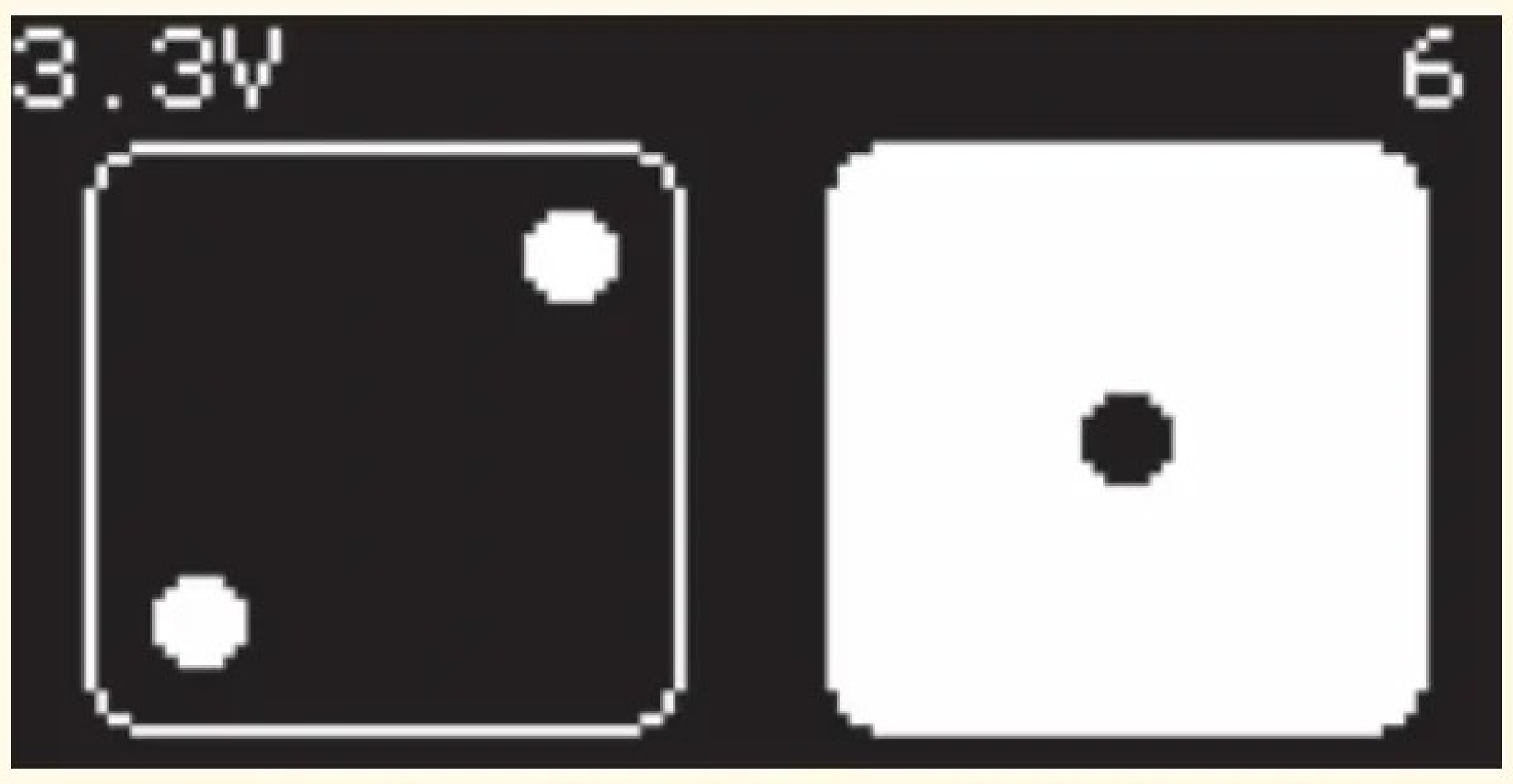
Configuration

Pressing S1 and S2 (BACK and OK) together will enter the SETTINGS mode. In this mode, the LEFT and RIGHT buttons cycle between the available parameters, while UP and DOWN will change them. Pressing BACK will exit SETTINGS.

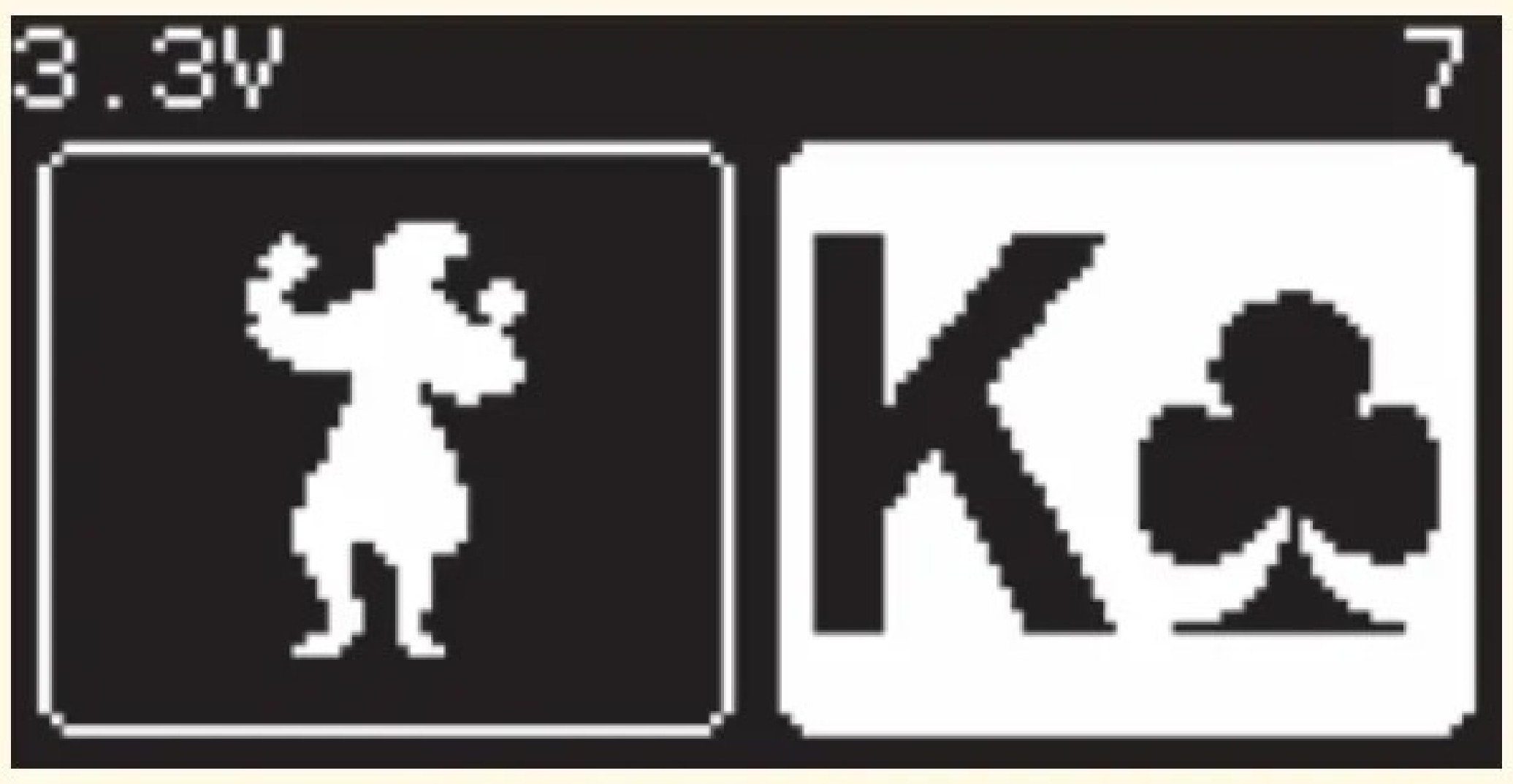
There are a handful of display and operation preferences, plus configuration for ten roll combinations that can be set up as you choose. They (and the other settings) are kept in EEPROM, so they take effect immediately and will be retained even if the battery is removed. Screens 4-12 and their accompanying captions explain each of



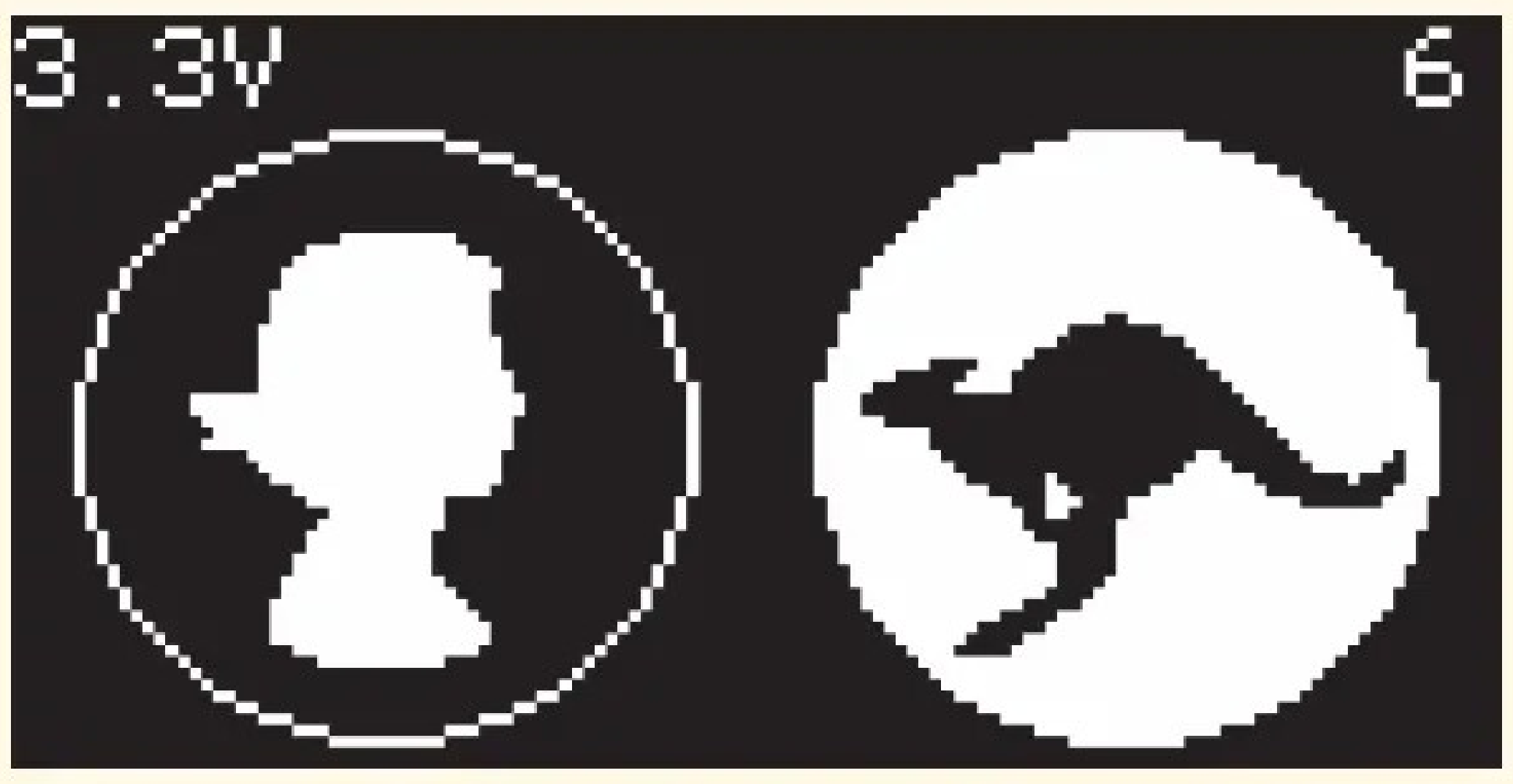
The PCB shown at left is only used as a 'panel' to protect the back of the main PCB, it has no components to solder to it.



Screen 13: dice with pips can show rolls up to nine. You'll note some nice touches, like the orientation of the two and three roll is not fixed but can vary, just like actual dice.



Screen 14: rolls (draws?) using the card options show the standard playing card symbols as seen here. Up to six Jokers can be added to the deck by selecting the 58-card option.



Screen 15: the coin toss shows its outcomes as images of an Australian penny, so it's well-suited for a traditional game of Two-Up.

the settings screens available on the FlexiDice.

Diagnostics

Screen 11 shows the noise multiplier diagnostic screen. It performs several rolls and reports on their fairness and correlation. Scores of 100% mean that the rolls are fair (equal number of 0s and 1s) and uncorrelated (any roll has an equal chance of following any other roll).

The result of all rolls is also delivered as serial (UART) data at 115,200 baud through pin 18 (RA1) of IC1, which is also pin 5 of the CON1 ICSP header (furthest from the > pin 1 marker). Pin 3 of CON1 is circuit ground. You could connect a USB-serial converter to these pins to dump this data into a computer for analysis.

All rolls from the main screens are also dumped via the serial port in this fashion. You can run repeated rolls on the main screen (for example, to accumulate numerous results on the serial port) by holding the DOWN button after pressing the UP button for a roll.

On Screen 11, values above 90% are typical and expected, although any one test might show a lower result. This is because any truly random phenomenon will occasionally show long runs of one particular value.

Running multiple tests will accumulate the results, and you should see the long-term results, which will be more representative. If you see values near 0%, the noise multiplier is not working and is probably stuck at a specific value. In that case, check the circuitry around IC2 and IC3, plus their connections to IC1.

During our research for this project, we came across an experiment (see https://pemag.au/link/abzz) where coins were tossed 40,000 times. It found that even a real coin shows a notable bias towards one result. Our tests would put the fairness and correlation of those real-life coin toss results at around 97%.

Completion

Once you are confident the FlexiDice is working as expected, you can secure the coin cell by using the M2 screw and nuts. Attach them to the hole near the cell to prevent it from being accidentally removed.

Add a dab of glue between the PCB and the vibration switch. This will reduce the strain on the leads.

Stick the double-sided foam tape to the back of the battery holder and use it to attach the panel PCB, aligning it with the main PCB. Some pads on the back of the panel PCB align to CON1, so you can solder some wires between the two for some extra stability if you want.

Play

Screens 13-15 show some of the different graphics the FlexiDice can display. They include dice 'pip' faces, playing cards and coin faces. For numbers higher than nine, numeric displays like those seen in Screen 2 must be used; they are also available for lower rolls.

Use the LEFT and RIGHT buttons to cycle between the various options, which include several single dice rolls and card picks and those that are configurable from SETTINGS.

Whenever the FlexiDice enters sleep mode, all settings are retained. Pressing any button apart from UP (or the vibration switch) will reinstate the previous display so that you can, for example, see what the last roll was.

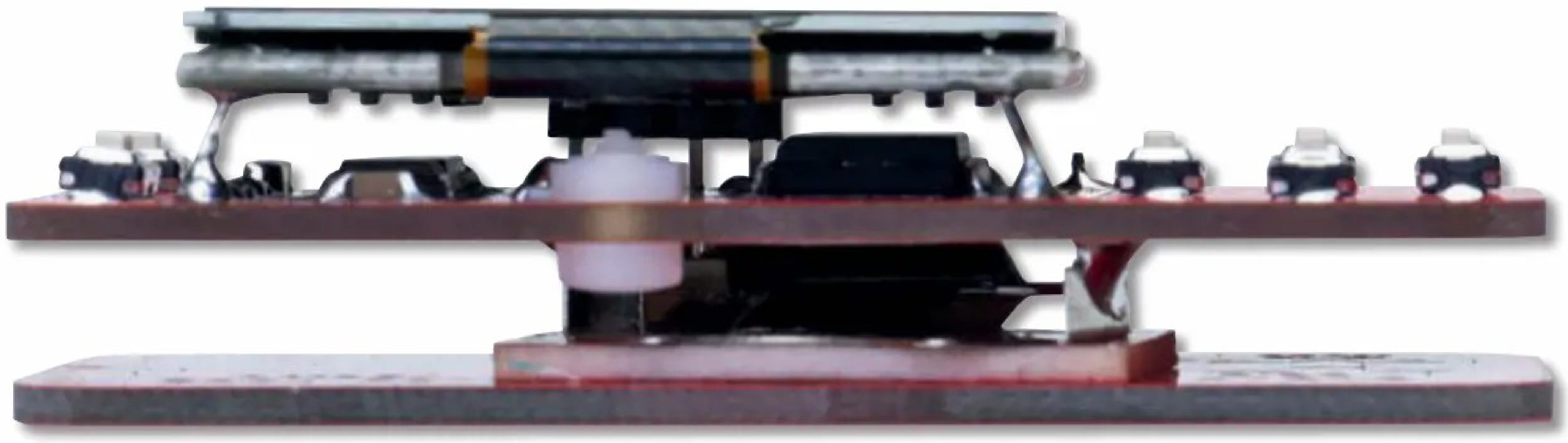
Pressing UP (or shaking) will always start a new roll, so there is no delay in getting a result after exiting sleep mode. This means it is less distracting when you are playing a game.

Because settings are saved in EEPROM, all the settings and presets will be retained even if you change the cell. The LFSR state is also saved every time the Dice goes to sleep, so there is less chance of the same result occurring repeatedly if you are using the LFSR.

Conclusion

The FlexiDice is a compact and handy substitute for all sorts of dice and can also be used to simulate coin tosses and card draws. You could even set it up to pick your numbers for Lotto draws.

It runs from a single coin cell and the cell voltage display should give you plenty of warning before it goes flat. It also looks like a tiny games console, so we hope some of our readers think of other playful applications for this hardware. PE



Double-sided foam tape is used to attach the protective panel PCB to the battery holder, although it isn't strictly required.

ETI BUNDLE (1) Teach-In 3, 4 and 5 – all on CD-ROM – only £18.95

ELECTRONICS TEACH-IN 3 – CD-ROM

Mike & Richard Tooley

ELECTRONICS TEACH-IN 3
CIRCUIT SURGERY
PRACTICALLY SPEAKING
INGENUITY UNLIMITED
FREE TEACH-IN 1 CD-ROM
TWO TEACH-INs FOR THE PRICE OF ONE!

The three sections of the *Teach-In 3* CD-ROM cover a huge range of subjects that will interest everyone involved in electronics – from newcomers to the hobby and students to experienced constructors and professionals.

The first section (80 pages) is dedicated to *Circuit Surgery*, *EPE/PE's* regular clinic dealing with readers' queries on circuit design problems – from voltage regulation to using SPICE circuit simulation software.

The second section – *Practically Speaking* – covers hands-on aspects of electronics construction. Again, a whole range of subjects, from soldering to avoiding problems with static electricity and identifying components is covered. Finally, our collection of *Ingenuity Unlimited* circuits provides over 40 circuit designs submitted by readers.

The CD-ROM also contains the complete *Electronics Teach-In 1* book, which provides a broad-based introduction to electronics in PDF form, plus interactive quizzes to test your knowledge and TINA circuit simulation software (a limited version – plus a specially written TINA Tutorial).

The *Teach-In 1* series covers everything from electric current through to microprocessors and microcontrollers, and each part includes demonstration circuits to build on breadboards or to simulate on your PC.

ELECTRONICS TEACH-IN 4 – CD-ROM
A BROAD-BASED INTRODUCTION TO ELECTRONICS

Mike & Richard Tooley

ELECTRONICS TEACH-IN 4
A BROAD-BASED INTRODUCTION TO ELECTRONICS
THE MODERN ELECTRONICS MANUAL

The *Teach-In 4* CD-ROM covers three of the most important electronics units that are currently studied in many schools and colleges. These include, Edexcel BTEC level 2 awards and the electronics units of the Diploma in Engineering, Level 2.

The CD-ROM also contains the full *Modern Electronics Manual*, worth £29.95. **The Manual** contains over 800 pages of electronics theory, projects, data, assembly instructions and web links.

A package of exceptional value that will appeal to anyone interested in learning about electronics – hobbyists, students or professionals.

ELECTRONICS TEACH-IN 5 – CD-ROM
JUMP START

Mike & Richard Tooley

ELECTRONICS TEACH-IN 5
JUMP START
PRACTICALLY SPEAKING
PIC N' MIX
TEACH-IN 2
TWO TEACH-INs FOR THE PRICE OF ONE!

15 design and build circuit projects for newcomers or those following courses in school and colleges.

The projects are: ■ Moisture Detector ■ Quiz Machine ■ Battery Voltage Checker ■ Solar-Powered Charger ■ Versatile Theft Alarm ■ Spooky Circuits ■ Frost Alarm ■ Mini Christmas Lights ■ iPod Speaker ■ Logic Probe ■ DC Motor Controller ■ Egg Timer ■ Signal Injector Probe ■ Simple Radio Receiver ■ Temperature Alarm.

PLUS

PIC'n' Mix – starting out with the popular range of PIC microcontrollers and *Practically Speaking* – tips and techniques for project construction.

The CD-ROM also contains:

- Complete *Teach-In 2* book, a practical introduction to PIC microprocessors
- MikroElektronika, Microchip and L-Tek PoScope software.

Three Teach-ins for the great price of £18.95

PLUS you also get the contents of the free CD-ROM from each issue...

... so that's another TWO Teach-Ins and The Full Modern Electronics Manual!

What a Bargain!!

ETI BUNDLE (2) Teach-In 6, 7 and 8 – all on CD-ROM – only £18.95

ELECTRONICS TEACH-IN 6 – CD-ROM
A COMPREHENSIVE GUIDE TO RASPBERRY Pi

Mike & Richard Tooley

Teach-In 6 contains an exciting series of articles that provides a complete introduction to the Raspberry Pi, the low cost computer that has taken the education and computing world by storm.

This latest book in our *Teach-In* series will appeal to electronics enthusiasts and computer buffs who want to get to grips with the Raspberry Pi.

Teach-In 6 is for anyone searching for ideas to use their Pi, or who has an idea for a project but doesn't know how to turn it into reality. This book will prove invaluable for anyone fascinated by the revolutionary Pi. It covers:

- Pi programming
- Pi hardware
- Pi communications
- Pi Projects
- Pi Class
- Python Quickstart
- Pi World
- ...and much more!

The *Teach-In 6* CD-ROM also contains all the necessary software for the series, so that readers and circuit designers can get started quickly and easily with the projects and ideas covered.

ELECTRONICS TEACH-IN 6
RASPBERRY Pi
A COMPREHENSIVE GUIDE TO RASPBERRY Pi

ELECTRONICS TEACH-IN 7 – CD-ROM
DISCRETE LINEAR CIRCUIT DESIGN

Mike & Richard Tooley

Teach-In 7 is a complete introduction to the design of analogue electronic circuits. It is ideal for everyone interested in electronics as a hobby and for those studying technology at schools and colleges. The CD-ROM also contains all the circuit software for the course, plus demo CAD software for use with the *Teach-In* series.

- Discrete Linear Circuit Design
- Understand linear circuit design
- Learn with 'TINA' – modern CAD software
- Design simple, but elegant circuits
- Five projects to build:
  - i) Pre-amp
  - ii) Headphone Amp
  - iii) Tone Control
  - iv) VU-meter
  - v) High Performance Audio Power Amp.

PLUS

Audio Out – an analogue expert's take on specialist circuits

Practically Speaking – the techniques of project building.

ELECTRONICS TEACH-IN 7
DISCRETE LINEAR CIRCUIT DESIGN

ELECTRONICS TEACH-IN 8 – CD-ROM
INTRODUCING THE ARDUINO

Mike & Richard Tooley

Hardware: learn about components and circuits
Programming: powerful integrated development system
Microcontrollers: understand control operations
Communications: connect to PCs and other Arduinos.

Teach-In 8 is an exciting series designed for electronics enthusiasts who want to get to grips with the inexpensive, popular Arduino microcontroller, as well as coding enthusiasts who want to explore hardware and interfacing. It will provide a one-stop source of ideas and practical information.

The Arduino offers a truly effective platform for developing a huge variety of projects; from operating a set of Christmas tree lights to remotely controlling a robotic vehicle through wireless or the Internet.

Teach-In 8 is based around a series of practical projects with plenty of information for customisation.

This book also includes *PIC n' Mix*: 'PICs and the PICkit 3 – A Beginners guide' by Mike O'Keefe and *Circuit Surgery* by Ian Bell – 'State Machines part 1 and 2'.

The CD-ROM includes the files for:

- *Teach-In 8*
- Microchip MPLAB IDE XC8 8-bit compiler
- PICkit 3 User Guide
- Lab-Nation Smartscope software.

ELECTRONICS TEACH-IN 8
INTRODUCING THE ARDUINO

ORDER YOUR BUNDLE TODAY!
JUST CALL 01202 880299 – OR VISIT www.electronpublishing.com

MAX'S COOL BEANS

By Max the Magnificent

WEIRD & WONDERFUL ARDUINO PROJECTS

Part 8: in which we handle switches like heroes

As usual, I have a cornucopia of cool concepts to convey, and I'm struggling to decide where to start. Now that I think about it, I recently came across some interesting nuggets of knowledge regarding the expression of integer values in different bases, so let's kick off there.

2B or not 2B

Currently, in the standard C programming language, integer values, such as literals and constants, can be written using either decimal or hexadecimal notation; other bases, such as binary, are not supported by default. Decimal values are expressed as standard numbers written without any prefix. For example:

```
int x = 42;
```

By comparison, hexadecimal values are prefixed with `0x` or `0X` (I prefer the former). For example, the hexadecimal equivalent of 42 in decimal could be expressed as follows:

```
int y = 0x2A;
```

In the case of the C++ programming language, the 2014 release (known as C++14) introduced support for binary values that are prefixed with `0b` or `0B` (I prefer the former). This means that the binary equivalent of 42 in decimal can be expressed as follows:

```
int z = 0b101010;
```

As fate would have it, the Arduino's integrated development environment (IDE) uses a combination of C and C++. As a result, the current version of the IDE supports decimal, hexadecimal and binary notations.

Don't get bitten by a byte

The original C language featured four integer data types: `char`, `short`, `int` and `long`. We've discussed these before. They can all be modified with the `signed` or `unsigned` keywords. The `short`, `int` and `long` types are signed by default; that is, `signed` is implied if this keyword is omitted.

In contrast, the `char` type is special. It's always eight bits (one byte) in size, but it may be signed or unsigned by default, depending on the compiler being used. As you can imagine, this has resulted in countless problems when programs are ported from one machine (and its compiler) to another.

When the Arduino was launched in 2005, its creators aimed to make things easy for beginners to grasp. As part of this, they added a new integer data type called `byte`, which is an unsigned 8-bit integer that can hold values in the range 0 to 255 in decimal. In reality, 'under the hood', `byte` is an alias for `unsigned char` (or, more technically, `uint8_t`).

The Arduino's documentation also states that we can use binary values in the form `Bxxxxxxxx` (only uppercase 'B' is allowed). I must admit that I always wondered why we are obliged to specify exactly eight bits.

For example, if we wish to specify a 6-bit binary value of 101010, then using an 8-bit `B00101010` (with two leading zeros) will work, while using a 6-bit `B101010` (without the leading zeros) will cause the compiler to 'throw a wobbly' (flag an error).

It turns out that, to support this capability ('under the hood' once again), the Arduino's cunning creators include a suite of definitions like the following:

```
#define B00000000   0
#define B00000001   1
#define B00000010   2
#define B00000011   3
…
#define B11111110 254
#define B11111111 255
```

So, when working with the Arduino, you can use the `byte` data type along with `Bxxxxxxxx` values if you wish. However, if you want to make your programs portable or if you plan on using other microcontrollers (and their compilers) in the future, stick to the fixed-width data types (`int8_t`, `uint8_t`, `int16t`, `uint16_t` etc) and the `0b` prefix for binary values.

Setting the scene

As you may recall, as part of our general experiments in the January 2025

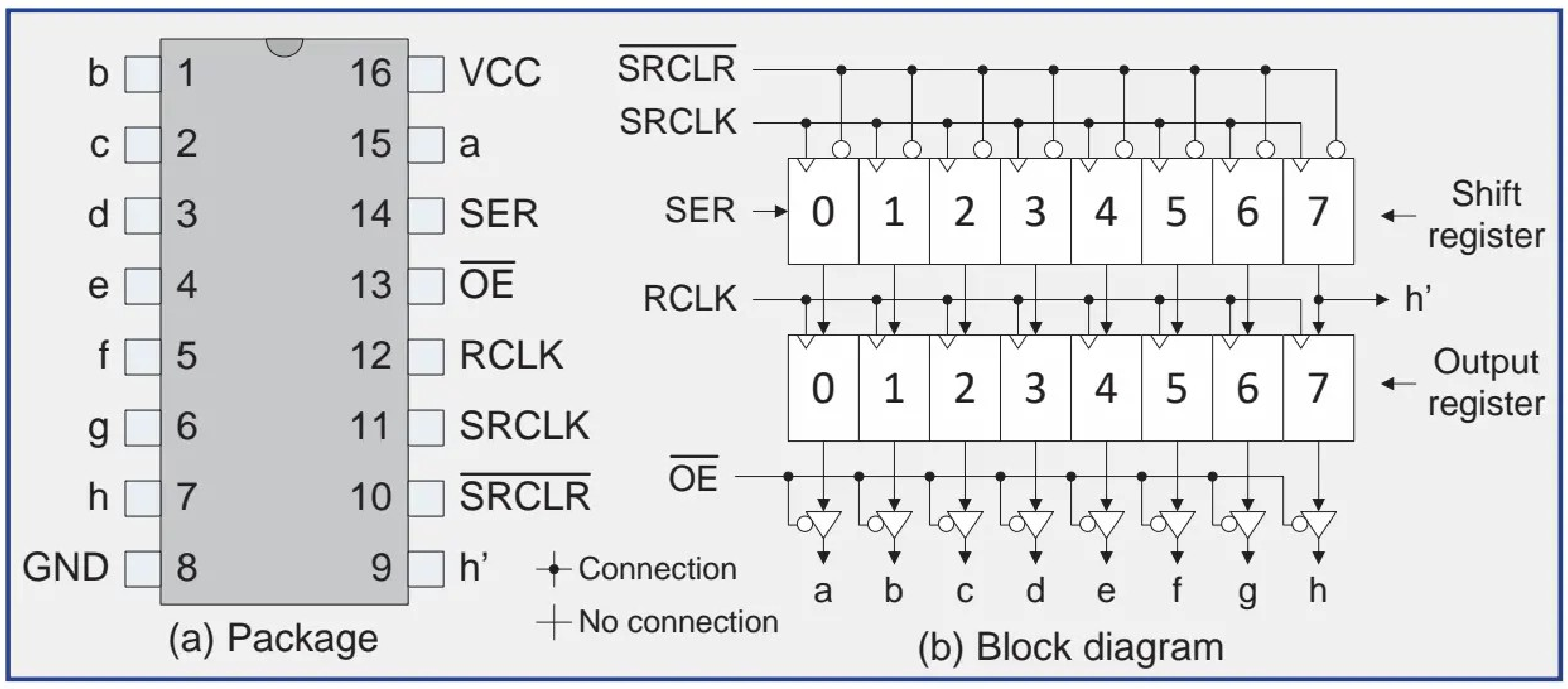


Fig.1: the 8-bit 74HC595 SIPO shift register IC configuration.

column, we populated a breadboard with an eight-segment light-emitting diode (LED) bar graph display, along with current-limiting resistors. At that time, we drove all the display segments directly using eight of our Arduino Uno R3 microcontroller's digital input/output (I/O) pins.

Later, in the June 2025 issue, we added an eight-bit serial-in, parallel-out (SIPO) shift register in the form of a 74HC595 integrated circuit (IC). We are using a breadboard-friendly dual-in-line package (DIP) version, as illustrated in Fig.1(a).

As per its data sheet (**https://pemag.au/link/ac1o**), the eight primary outputs from this device are named **a** to **h** (or Q_A to Q_H). However, we can also think of them as being numbered 0 to 7. We moved to using these signals to drive the segments on our display.

In last month's column, we utilised our 74HC595 with a 2-input XNOR function (which we created using a 2-input XOR and a NOT gate) to implement a linear feedback shift register (LFSR), which generates a pseudorandom sequence of values. We will return to this in next month's column; for now, we have other fish to fry.

As illustrated in Fig.1(b), the 74HC595 actually contains two 8-bit registers: the shift register itself and an output register.

In addition to the **SER** (serial data) input, this device has four control signals. However, we are currently using only three of these because we are connecting the **$\overline{\text{OE}}$** (output enable) input to ground, permanently enabling the outputs.

Regarding the remaining control signals, the active-low **$\overline{\text{SRCLR}}$** (shift register clear) input, which is level-sensitive, clears all the shift register bits to 0. By comparison, a rising edge (a 0-to-1 transition) on the positive-edge-triggered **SRCLK** (shift register clock) input performs a 1-bit shift on the current contents of the shift register.

As part of this, the value being presented to the **SER** input is copied into bit 0; the original contents of bit 0 are copied into bit 1; the original contents of bit 1 are copied into bit 2, and so on. We can visualise the original contents of bit 7 as conceptually 'falling off the end'.

Once we've changed the contents of the shift register via its **$\overline{\text{SRCLR}}$** or **SRCLK** inputs, we need to apply a rising edge (a 0-to-1 transition) to the positive-edge-triggered **RCLK** (output register clock) input. This causes the current contents of the shift register to be copied into the output register, thereby making them visible to the outside world.

Bouncy, bouncy

In our June 2025 column, we added three momentary pushbutton switches to our breadboard and used them to directly drive the 74HC595's **$\overline{\text{SRCLR}}$**, **SRCLK**, and **RCLK** inputs. Almost immediately, we ran into switch bounce problems.

As we discussed at that time, when we activate or deactivate an electromechanical switch, it rarely executes a single, clean transition. Instead, it can bounce up to 100+ times over a period as long as six milliseconds (6ms).

Switch bounce isn't a concern if we're doing something simple, such as using a switch to control LED(s), because the bounces are faster than we can perceive. However, it can cause problems when using the switch to drive high-speed digital logic, such as our shift register or the input to a microcontroller, because each bounce can be perceived as a legitimate switching event.

In our case, pressing the **SRCLK** switch caused the shift register to be clocked multiple times.

There are various methods for debouncing signals from switches, with both hardware and software techniques. The solution we adopted was to feed the outputs from the switches into our Arduino, debounce the switches in software, and then feed the debounced versions to the shift register (Fig.2).

We began by treating each switch independently. Every time we pressed the **$\overline{\text{SRCLR}}$** or **SRCLK** switches to modify the contents of the shift register, we had to press the **RCLK** switch to copy these contents into the output register. To prevent our switch-pressing finger from becoming fatigued, we modified our code to automatically add an **RCLK** pulse following an **$\overline{\text{SRCLR}}$** or **SRCLK** event.

You can download a copy of the latest and greatest version of our current switch debounce code and see the current state of our breadboard in the files named **CB-aug25-code-01.txt** and **CB-aug25-brd-01.pdf**, respectively. As usual, all files mentioned in this column are available as part of the August 2025 download package from the *PE* website at **https://pemag.au/link/ac6f**

They bounce how long?!

Switch bounce is a contentious topic. Many references state that the bouncing lasts no longer than 1ms. The reason I've stated 6ms is that several years ago, one of my friends, an acknowledged expert in embedded system design, performed extensive tests on a wide variety of switches, finding an average bounce time of around 2ms, and a maximum bounce time of around 6ms.

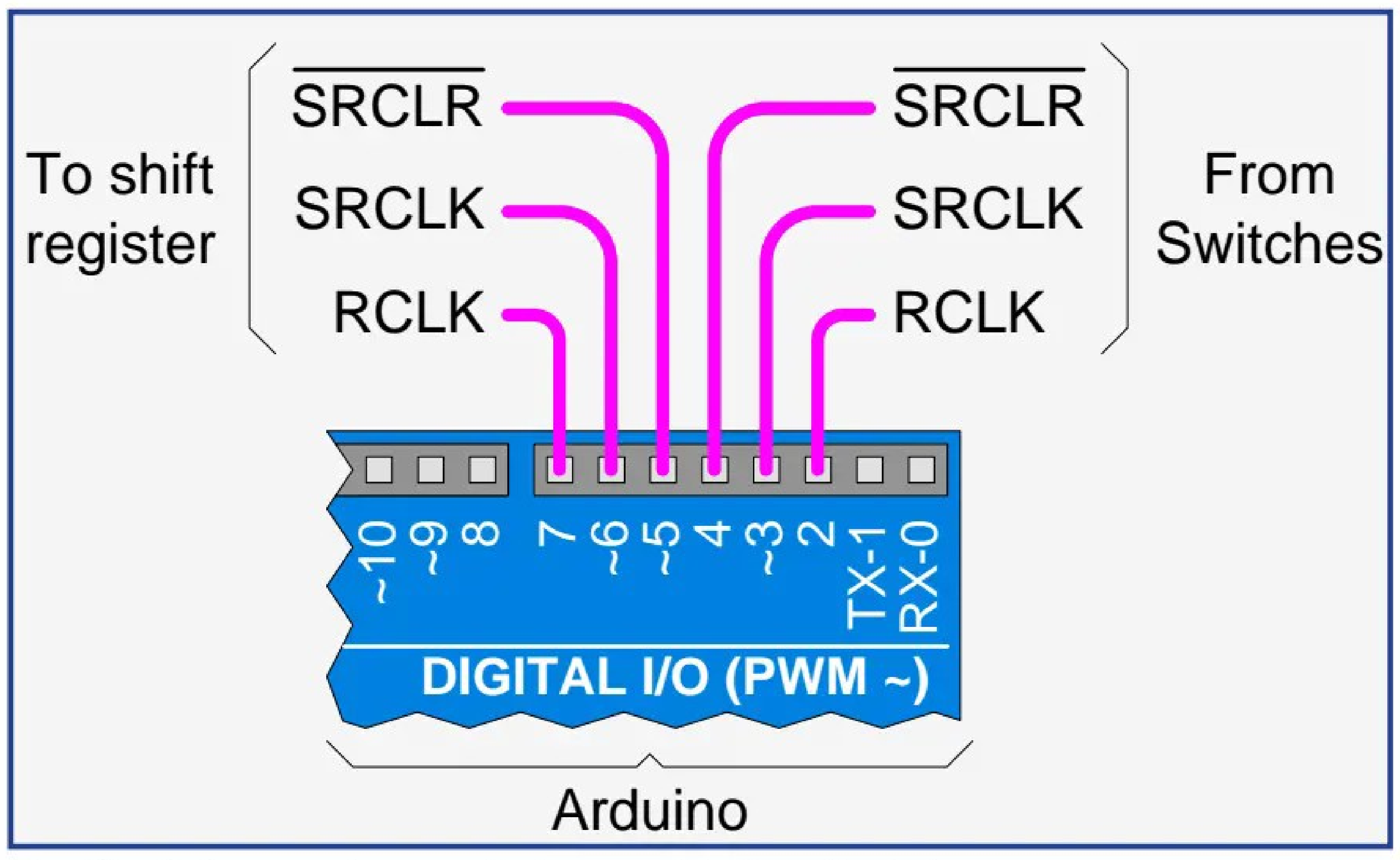


Fig.2: debouncing switches with the Arduino.

As we discussed in last month's column how, in less critical deployments, such as those in consumer electronics, designers often assume that worst-case bounce durations are as low as 1ms, depending on the switch type and the application.

Does pressing a button on your remote control sometimes cause your TV to jump multiple channels? Now you know why!

In the case of embedded systems and microcontroller applications, designers typically assume a worst-case bounce durations of 10ms. Meanwhile, designers of mission-critical and safety-critical systems tend to be conservative, so they assume worst-case bounce durations of 20ms.

In my own debounce solutions, I assume the actual real-world bounce duration to be no more than 6ms, but I usually design to accommodate longer bounce durations 'just in case'.

But what about…

There are many shortfalls in our current 'cheap and cheerful' switch handling solution. Let's start with the low-hanging fruit. As we've discussed in previous columns, the 74HC595 doesn't contain any special initialisation logic. This means it can power up containing a random collection of 0s and 1s.

Thus far, we've addressed this concern by commencing every experiment by hand-clearing the device. The great thing about having the Arduino in the circuit is that we can tweak our **setup()** function to do this for us.

Another problem with our current code is that it assumes that all the pushbutton switches will be in their inactive states when power is first applied to the system. In the real world, it's not beyond the bounds of imagination that the user may press and hold one or more switches prior to powering up the system, so it behooves us to take that into account.

Mission-critical and safety-critical systems must be as foolproof as possible. Of course, we should never forget what the late, great Douglas Adams wrote about this: "A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."

As a counterpoint to this, when I give a lecture on embedded systems, one of the points I usually make is, "Half of the safety systems we build into a system are there to tell us when the user has turned the other half off."

We have multiple options here. For example, on power-up, in our **setup()** function, we could have the Arduino to read the current states of the switches and write these values to our shift register.

Another possibility – one we might consider using in the retro game console we are building – would be to enter a special configuration mode if we detect that one of our switches is active when the system is powered on. And another alternative would be to simply wait for all the switches to be in (or return to) their inactive states before proceeding to the main body of the program.

Sad to relate, though, that we still haven't addressed the main problem, which is that…

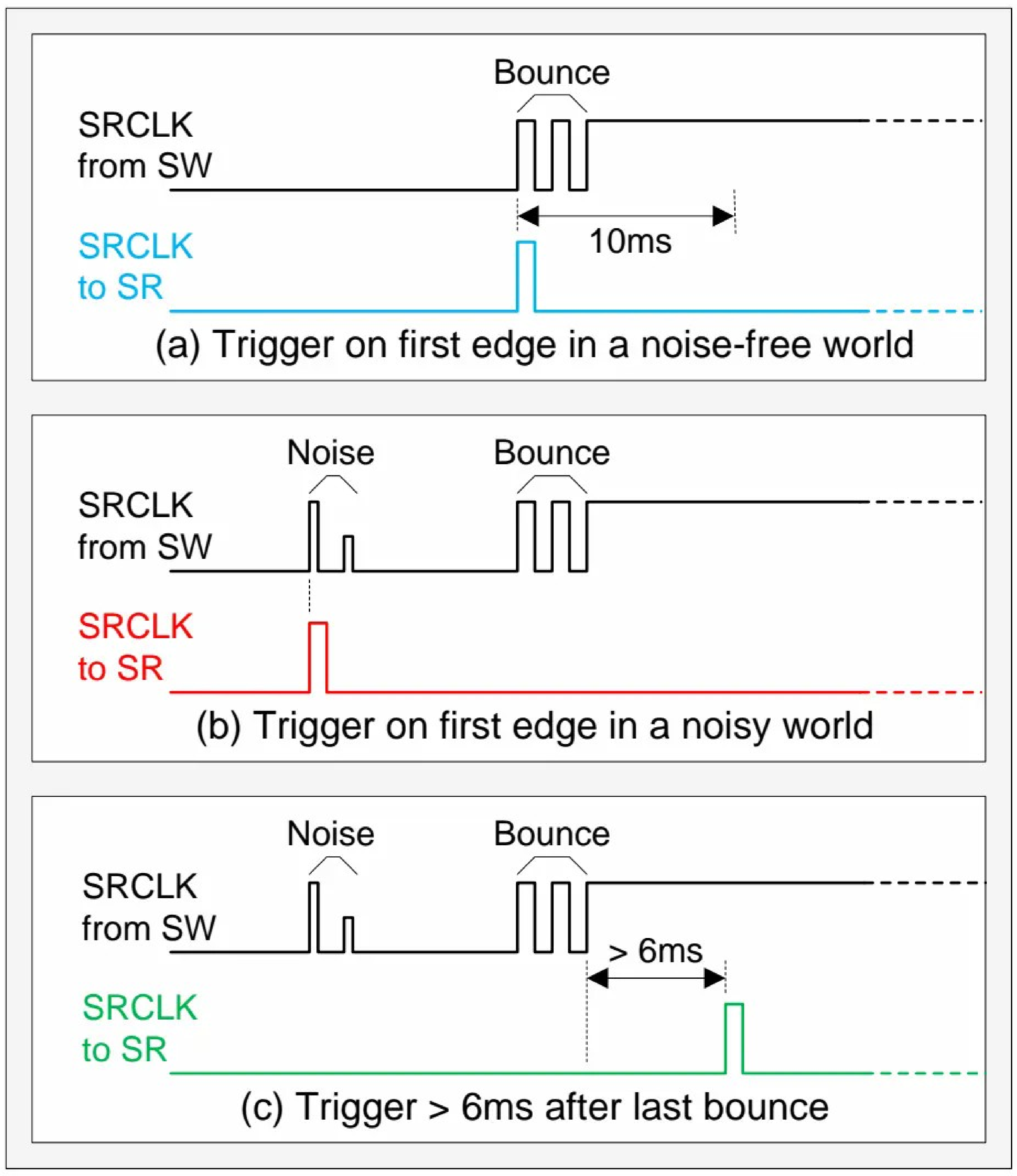
We live in a ~~beautiful~~ noisy world

When we introduced our initial switch debounce solution in the June 2025 issue, I wrote, "In an ideal world, starting from the switch being in its inactive state, the Arduino would detect the leading edge of the activation and cause its output to the shift register (SR) to respond immediately." This is essentially how we implemented things.

However, I also coyly noted that, "This approach may not be quite as ideal as we might suppose."

Consider our current approach, as illustrated in Fig.3(a). As soon as we see the first edge of what we assume to be the start of a switch transition on the **SRCLK** signal from the switch, we generate a pulse on the **SRCLK** signal to the shift register (we follow this with a pulse on the **RCLK** signal to the shift register, but we aren't showing that here to keep things simple).

If you examine our code, you'll see that we then use a **delay(10)** statement to wait 10 milliseconds, thereby allowing any bouncing to occur before we return to checking the switch. 10ms is a nice round number, and it's greater than the 6ms worst-case bouncing we might expect to see.


Bounce
SRCLK from SW
10ms
SRCLK to SR
(a) Trigger on first edge in a noise-free world
Noise
Bounce
SRCLK from SW
SRCLK to SR
(b) Trigger on first edge in a noisy world
Noise
Bounce
SRCLK from SW
> 6ms
SRCLK to SR
(c) Trigger > 6ms after last bounce
Fig.3: strategies for addressing noisy digital signals.

The **delay()** function is blocking, which leaves the Arduino twiddling its metaphorical thumbs; that is, it prevents us from doing anything useful, such as reading new values from the Arduino's inputs, performing computations, and driving new values to its outputs.

The Arduino Uno R3 we are using in these experiments runs at 16MHz (16 million clock cycles per second). This means that doing nothing for 10ms equates to 160,000 clock cycles that could be employed to do something useful.

However, the real problem with our existing solution is that it doesn't account for noise in the form of electromagnetic interference (EMI). This can originate from natural sources, such as lightning, or human-made sources, including radio transmitters or motors switching on and off.

With our program responding to the first active edge on the **SRCLK** signal, consider what would happen if noise were present on this signal, as depicted in Fig.3(b). That's right, our current code would 'jump the gun', as it were. This would be, ahem, unfortunate if someone were using this code as part of something like a missile launch control system.

To be honest, EMI is probably not something we need to worry about in the case of our home-based experiments, but it is a significant concern in mission-critical and safety-critical applications such as industrial automation, medical devices, and aerospace.

And, of course, one of our goals in these columns is to learn and apply 'best practices', enabling the knowledge gained to be deployed in real-world applications.

It may be worth reminding ourselves that the timing relationships in the waveforms depicted in our illustrations, such as Fig.3, are not to scale. For example, we might take a coffee break or go for a long walk between adjacent presses of our **SRCLK** switch, which means the noise could occur seconds, minutes, hours (even days) before someone presses the switch.

So, what we're going to do, as illustrated in Fig.3(c), is wait until the signals from our switches have been steady for more than 6ms following the last bounce (thereby ensuring that we've really and truly stopped bouncing) before taking any action. This also means that brief pulses from EMI or other sources of electrical noise will be ignored.

Tortuous terminology

Before we proceed, there's something else that deserves mention. The **SRCLK** input to our shift register is described as "edge-triggered" because all the action occurs at an edge or transition. In this case, we're talking about positive edges in the form of 0-to-1 transitions.

By comparison, the **$\overline{\text{SRCLR}}$** input to our shift register is said to be "level-triggered" (or "level-sensitive") because it's the active level of this signal that's important. In the case of our shift register, if the **$\overline{\text{SRCLR}}$** input is low state, the **SRCLK** input will have no effect on the register's contents.

A 'synchronous system' is a digital system in which a shared clock signal coordinates all operations. This signal, typically in the form of a square(ish) wave, provides a regular timing reference, and all changes in the system's state occur in sync with the clock's edges.

In our case, even though we're using a hand-pressed switch to instigate our clock signal, this input still defines

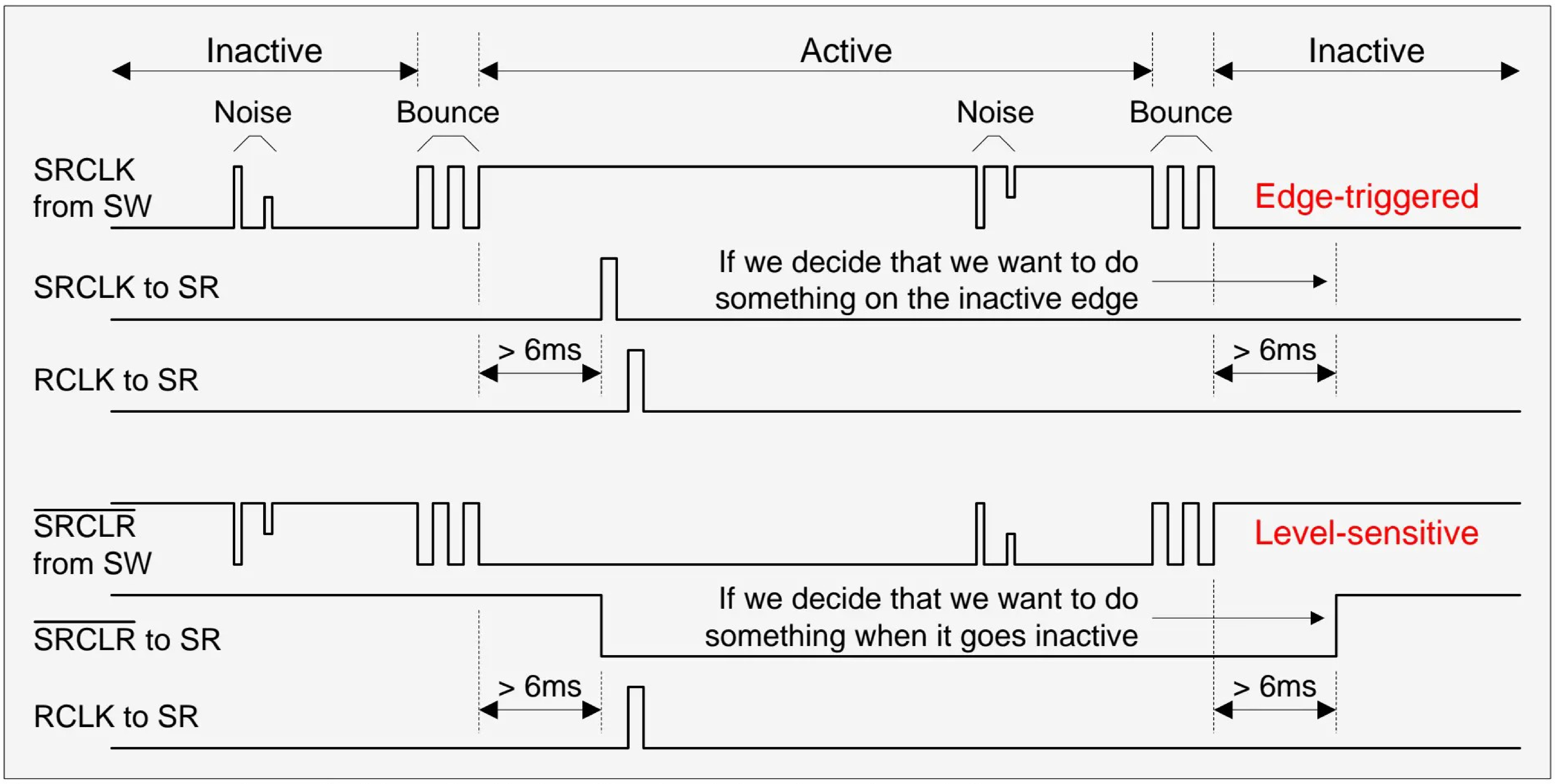

Inactive
Active
Inactive
Noise
Bounce
Noise
Bounce
SRCLK from SW
Edge-triggered
SRCLK to SR
If we decide that we want to do something on the inactive edge
RCLK to SR
> 6ms
> 6ms
$\overline{\text{SRCLR}}$ from SW
Level-sensitive
$\overline{\text{SRCLR}}$ to SR
If we decide that we want to do something when it goes inactive
RCLK to SR
> 6ms
> 6ms

Fig.4: our newer and more flexible method for handling signals from bouncing switch contacts.

the synchronous domain of the shift register. That is, whatever happens only on a clock edge is synchronous to that edge, even if the clock is erratic or driven by a human.

Meanwhile, the **$\overline{\text{SRCLR}}$** input, being level-sensitive and immediate, is said to be 'asynchronous' to that clock domain. That is, it takes effect independently of the clock.

Switch handling revisited

Our original switch-handling implementation primarily focused on what would happen when our switches were pressed, but we didn't spend much time worrying about what to do when they were released. This wasn't a problem in this case, but some systems may require us to support different actions when our switches are pressed and released.

Bearing all this in mind, let's create a solution capable of handling all the cases illustrated in Fig.4.

For this portion of our discussions, let's assume our control signals power up in their inactive states. Let's start with the **SRCLK** signal coming from the switch. Our routine will ignore any noise on this signal, responding only to an active edge when we know the signal has stopped bouncing (ie, it is stable for more than 6ms).

At this point, it will first apply a positive-going pulse to the **SRCLK** signal feeding the shift register, followed by a positive-going pulse on the **RCLK** signal feeding the shift register. Additionally, we will design our routine to allow us to perform actions on this signal's inactive edge if needed in the future.

Next, consider the **$\overline{\text{SRCLR}}$** signal coming from the switch. In our original code, we used this to first apply a negative-going pulse to the **$\overline{\text{SRCLR}}$** signal feeding the shift register, followed by a positive-going pulse on the **RCLK** signal feeding the shift register.

We could follow the same approach in our new routine. However, since I want to demonstrate alternative ways of doing things, I've decided that we will treat the **$\overline{\text{SRCLR}}$** signal feeding the shift register as a debounced version of the **$\overline{\text{SRCLR}}$** signal coming from the switch.

Counters vs shift registers

Let's take a step back and contemplate things at a high level of abstraction. What we really want is a single solution that can be used to debounce any of our switches, regardless of whether we wish their signals to operate in an edge-triggered or level-sensitive manner. Our solution should be capable of handling both positive and negative edges, as well as both active-low and active-high signals.

As part of this, we will monitor the signals from our switches, wait for them to transition from one state to another, then wait for 16ms after the final bounce before performing any actions.

Why 16ms? Well, it's significantly longer than the worst-case 6ms of expected switch bounce; it sits comfortably between the 10ms commonly used in embedded systems and the 20ms typical for mission- and safety-critical applications; and it integrates well with our implementation strategy, as discussed below.

A very common software technique is to use a loop that executes periodically (say, once every millisecond) in conjunction with a counter.

Suppose a signal from a switch starts off low. In this case, all we need to do is wait for it to be continuously high for 16ms. One way to achieve this is to initialise our counter with a value of 16. Every time our loop executes, we will check the state of the switch. If it's high, we decrement the contents of the counter; if it's low, we reload the counter with a value of 16.

When the counter reaches 0, we know that the signal has been steadily high for 16ms. Once we know the signal has transitioned to a stable high level, we can perform any required actions, then start waiting for it to transition low again.

Once again, we will initialise our counter with a value of 16. Once again, every time our loop executes, we will check the state of the switch. In this case, if it's low, we decrement the contents of the counter; if it's high, we reload the counter with a value of 16. When the counter reaches 0, we know that the signal has remained steadily low for 16ms.

Another approach involves utilising a software shift register; this is what we will do. We will use a 16-bit (two-byte) variable to store the recent 'history' of our shift register, where these 16 bits correspond to 16 readings over 16ms (ie, one per millisecond).

Suppose the signal from our switch starts off low. We can initialise our shift register with 0x0000 in hexadecimal (which is equivalent to sixteen 0s in binary). Every time our loop executes,

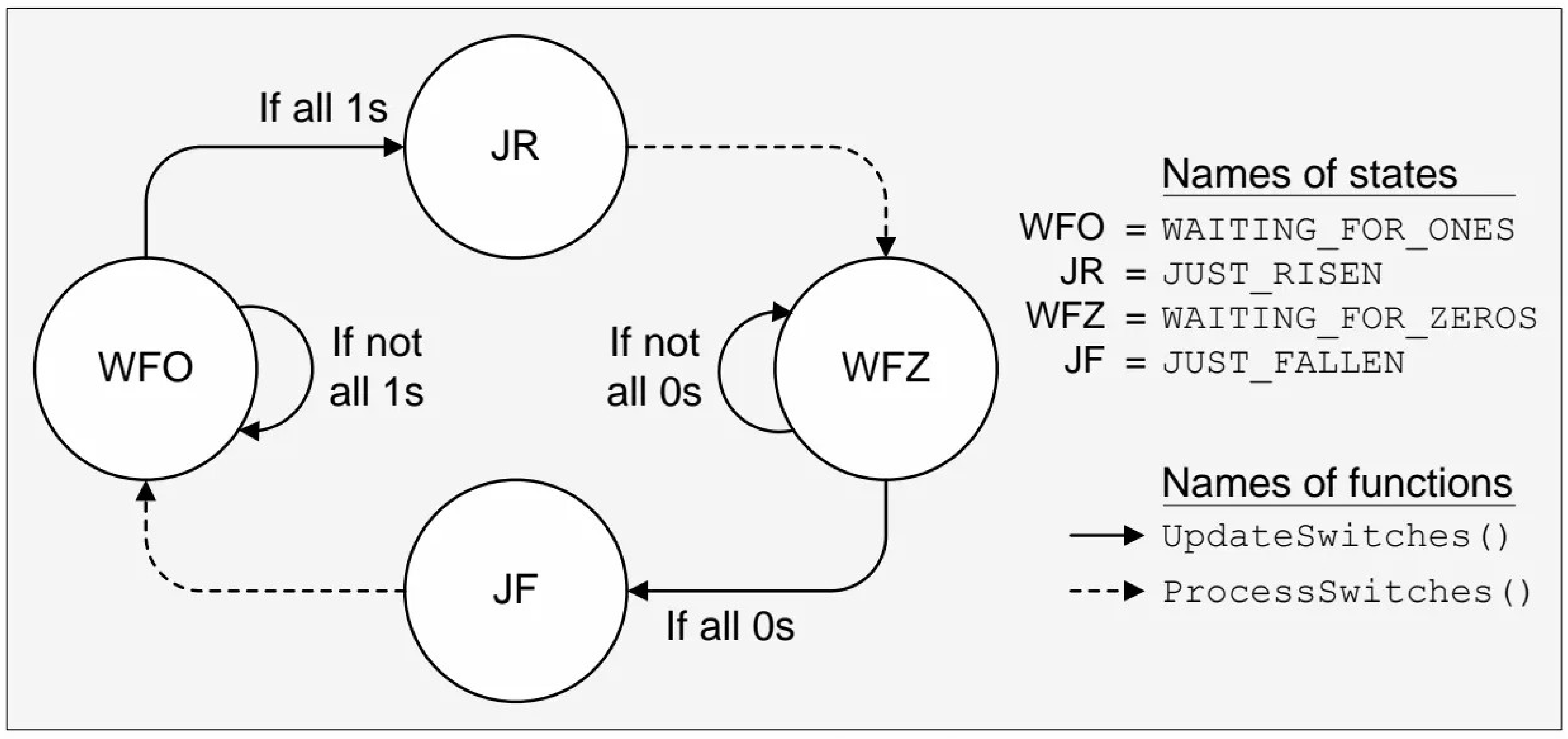


Fig.5: a state machine view of our switch handling solution.

we shift the register one bit to the left and then insert the current value (low = 0, high = 1) from the switch into the least significant bit.

When the shift register contains 0xFFFF (which is equivalent to sixteen 1s in binary), we know that the signal has been steadily high for 16ms. Once this happens, we can perform any required actions, then start waiting for it to transition back to all zeroes. In this case, we know the shift register already contains all 1s, so no initialisation is required.

As before, every time our loop executes, we shift the register one bit to the left and then insert the current 0 or 1 value from the switch into the least significant bit. When the shift register contains 0x0000, we know that the signal has been steadily low for 16ms.

Did you see that?

You may be thinking that 22ms – the sum of our 6ms worst-case bounce scenario and our 16ms delay after the final bounce – is a long time to wait for something to occur after we press a switch.

It's true that the human perception of delay in response to pressing a button can be surprisingly sensitive. However, there's a practical threshold below which most people won't consciously notice a delay.

A delay under 10ms is typically imperceptible to humans. A delay of 10-30ms may be slightly noticeable to very sensitive users, especially in contexts that require fast feedback (eg, gamers or musicians), but it generally still feels instantaneous. A delay of 30-100ms becomes noticeable to the average person, who may describe the system as feeling "laggy" or "slow to respond".

Most users will perceive any delay greater than 100ms, feeling the system is "sluggish" or "unresponsive". So, as a rule of thumb, we should aim to keep any button-to-action delays under 50ms. However, don't take my word for any of this, as this is a test you can easily perform for yourself using your Arduino, pushbuttons and LEDs.

Getting into a state

As previously discussed, we will employ a software-based switch debounce approach, using a shift register as part of our overall switch-handling solution.

The easiest way to implement our switch handling scheme is as a state machine (Fig.5). Each of our switches will have a corresponding state machine, and each state machine will have four states: **WFO** (waiting for ones), **JR** (just risen), **WFZ** (waiting for zeros), and **JF** (just fallen).

If the signal from a switch starts off at 0 when the system is powered up, we will load its corresponding shift register with 0x0000 and initialise its state machine to the **WFO** state. Contrariwise, if the signal from a switch starts off at 1, we will load its shift register with 0xFFFF and initialise its state machine to the **WFZ** state.

Every time we cycle around our 1ms loop thereafter, we will call two functions. First, we will call the **UpdateSwitches()** function, followed by the **ProcessSwitches()** function.

For each switch, the **UpdateSwitches()** function shifts that switch's register one bit to the left and adds the current state of the switch to the least-significant bit of the register.

Next, it performs a test. If the switch is currently in its **WFO** state and the shift register contains all 1s, it will transition to the **JR** state. Alternatively, if the switch is currently in its **WFZ** state and the shift register contains all 0s, it will transition to the **JF** state.

Now, this is the clever bit (well, I think so). For each switch, the **ProcessSwitches()** function checks to see if it's in its **JR** state, in which case it performs any necessary actions and then transitions the switch to its **WFZ** state, or if it's in its **JF** state, in which case it performs any necessary actions and then transitions the switch to its **WFO** state.

Consider the code

Remembering that I'm a rough-and-tough hardware design engineer (not a weedy software weenie), I must admit that I'm rather proud of this solution, which you can peruse and ponder in the file named **CB-aug25-code-02.txt**. I won't wear out my welcome by walking through this code line by line. Instead, we'll just hit some of the highlights.

First, we define a constant called **TICK** and assign it a value of 1; this will control our 1ms loop. We also create a bunch of definitions for the names of our control signals, the names of our state machine's states, and the values we will use to preload and compare our switch-related shift registers (**ALL_ZEROS** is defined as 0x0000, while **ALL_ONES** is defined as 0xFFFF).

The first part that's a little different from what we've seen before occurs on line 30, as illustrated in Listing 2(a). Remember that we are using the convention that the listing number (2 in this case) corresponds to the numerical part of the matching code file name ("02" here).

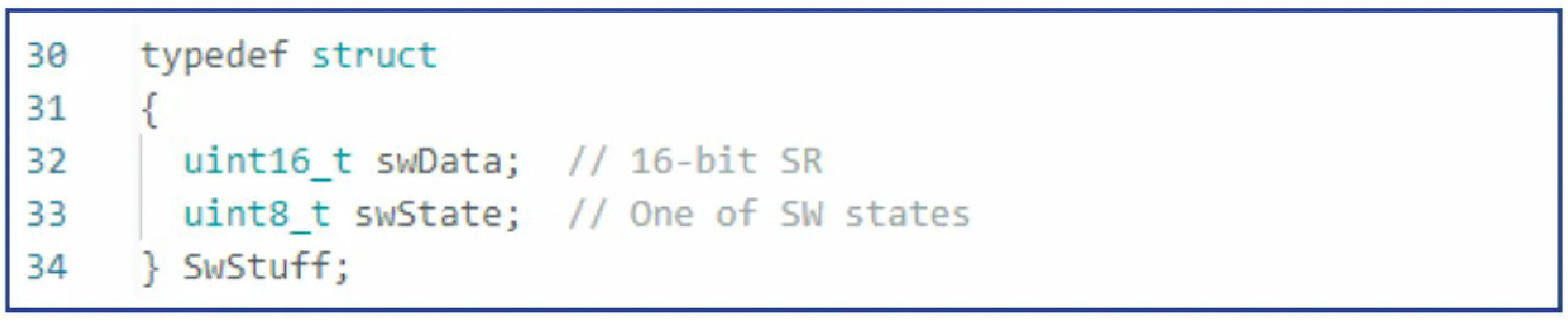
This code snippet defines a new type that combines two related pieces of data: **swData** (switch data), which will serve as our 16-bit shift register, and **swState** (switch state), where we will store the current state of the switch. The name of this structure is **SwStuff**.

We will use this new type to store the data associated with our state machines. As a reminder, we introduced the concepts of **struct** (short for "structure") and **typedef** (short for "type definition") in our *Coding Tips and Tricks* discussions in the December 2020 issue.

Later, on line 47, we declare an array of type **SwStuff** called **Switches[]**, as illustrated in Listing 2(b). In this case, **NUM_SIGNALS** (the number of control signals, which corresponds to the number of switches) has been defined as 3, but this can be easily changed to accommodate more or fewer switches.

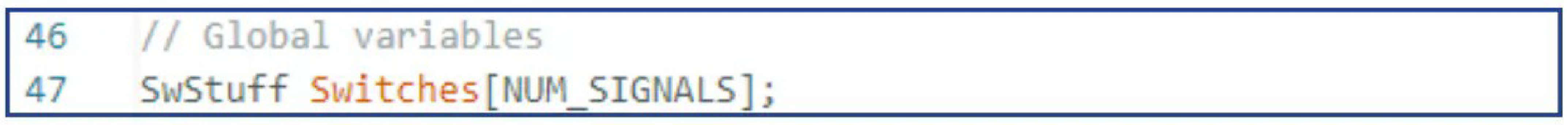
Now let's consider the **setup()** function. After configuring input and output pins and setting the signals to the shift register to their inactive states, we clear the shift register by pulsing its **SRCLR** signal, followed by its **RCLK** signal.

Next, we initialise the state machines associated with the switches by loading the shift register and setting the state for each switch. We then wait until all the switches are



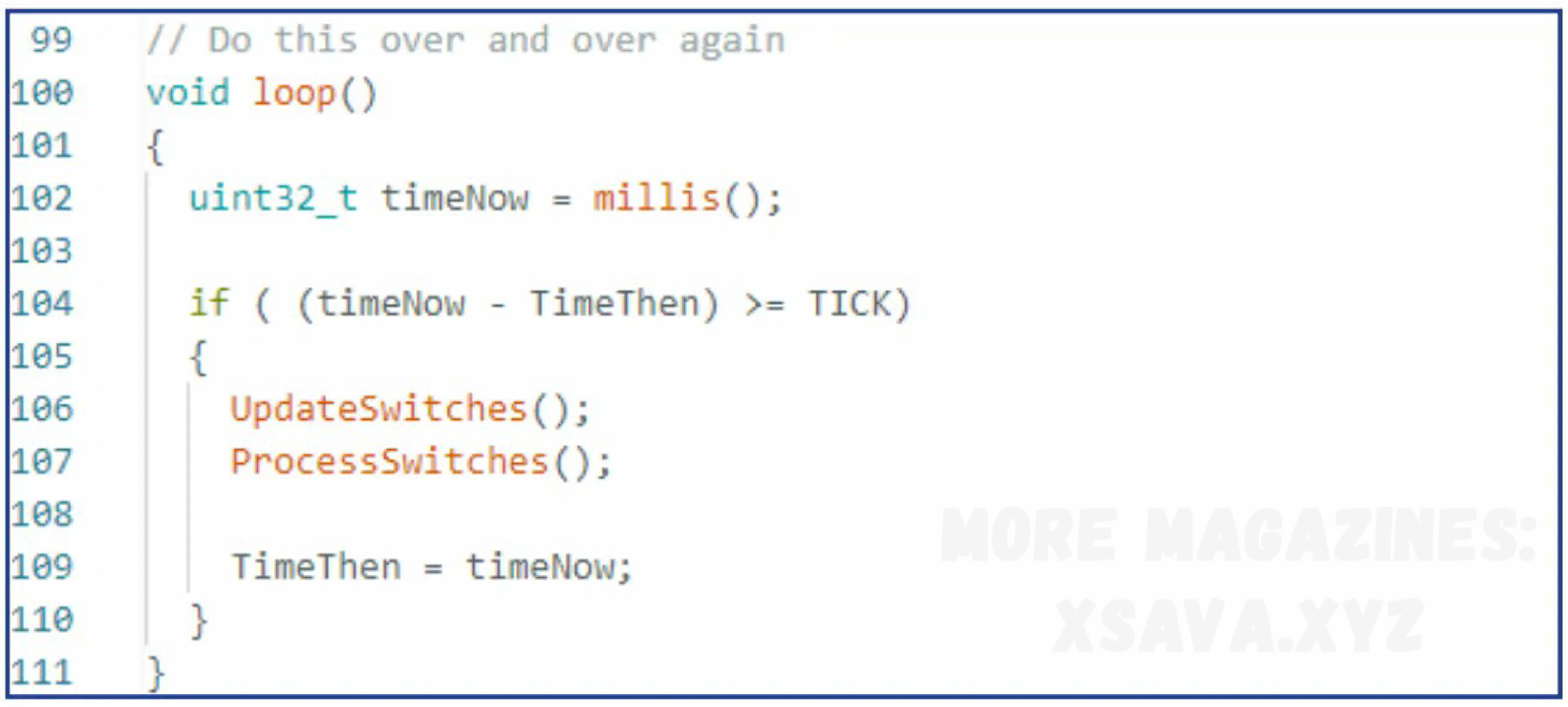
```
typedef struct
{
  uint16_t swData;   // 16-bit SR
  uint8_t swState;   // One of SW states
} SwStuff;
```

Listing 2(a): defining a new type.



```
// Global variables
SwStuff Switches[NUM_SIGNALS];
```

Listing 2(b): using our new type.



```
// Do this over and over again
void loop()
{
  uint32_t timeNow = millis();

  if ( (timeNow - TimeThen) >= TICK)
  {
    UpdateSwitches();
    ProcessSwitches();

    TimeThen = timeNow;
  }
}
```

MORE MAGAZINES:
XSAVA.XYZ

Listing 2(c): the main loop() function.

in their inactive states. If any of the switches are in their active states, we flash the Arduino's onboard LED (the one attached to pin 13) until they are returned to their inactive states.

You can test that this works by pressing one or more of the switches while the Arduino is powered up.

The **loop()** function is easy peasy lemon squeezy, as illustrated in Listing 2(c).

The Arduino's **millis()** function returns the number of milliseconds since our program started running. In this case, **TimeThen** is a global variable that we use to remember the last time an action was performed, while **timeNow** is a local variable that stores the current time.

In a nutshell, once every millisecond, we first call the **UpdateSwitches()** function and then call the **ProcessSwitches()** function. The great thing about this is how things are separated out. The **UpdateSwitches()** function treats each switch identically. It's only the **ProcessSwitches()** function that is required to handle any switch-specific actions.

We've already discussed what these functions do, so I'll leave it to you to review the code to see how everything plays together. I suggest that you peruse and ponder the program while comparing it to the actions depicted in Figs. 4 and 5. Please feel free to email me if you've got any questions (my address is at the end of this column).

Lastly, load this code into your Arduino and ensure that your LFSR continues to function as before, secure in the knowledge that everything will continue to work in the noisiest of noisy environments.

Consider the console

You may be thinking that we've put a lot of effort into our new switch handling solution, not least that the old one was 'good enough for government work', as they say. Still, there's a reason behind our madness.

Mad max, ready to power up the Arduino!

I know we've wandered a little off the beaten path, but we are still working toward building our retro games console, as seen in Photo 1.

Of particular interest here are the eight pushbutton switches mounted on the front panel: four white direction control switches and four coloured action control switches. Each of our games will use these switches in different ways, and our latest-and-greatest switch-handling solution can be easily enhanced to address this.

Next time

Give me strength! Once again, we've run out of time. There's so much to do and so little time. I'm too young for all this excitement!

In our next column, we will begin by addressing the homework problems posed in last month's column. After that, we will wire up our game console switches and apply our new switch handling code to these little scamps.

Meanwhile, if you have any thoughts that you'd care to share on anything you've read here, please feel free to drop me an email at max@clivemaxfield.com. And, as always, have a good one!

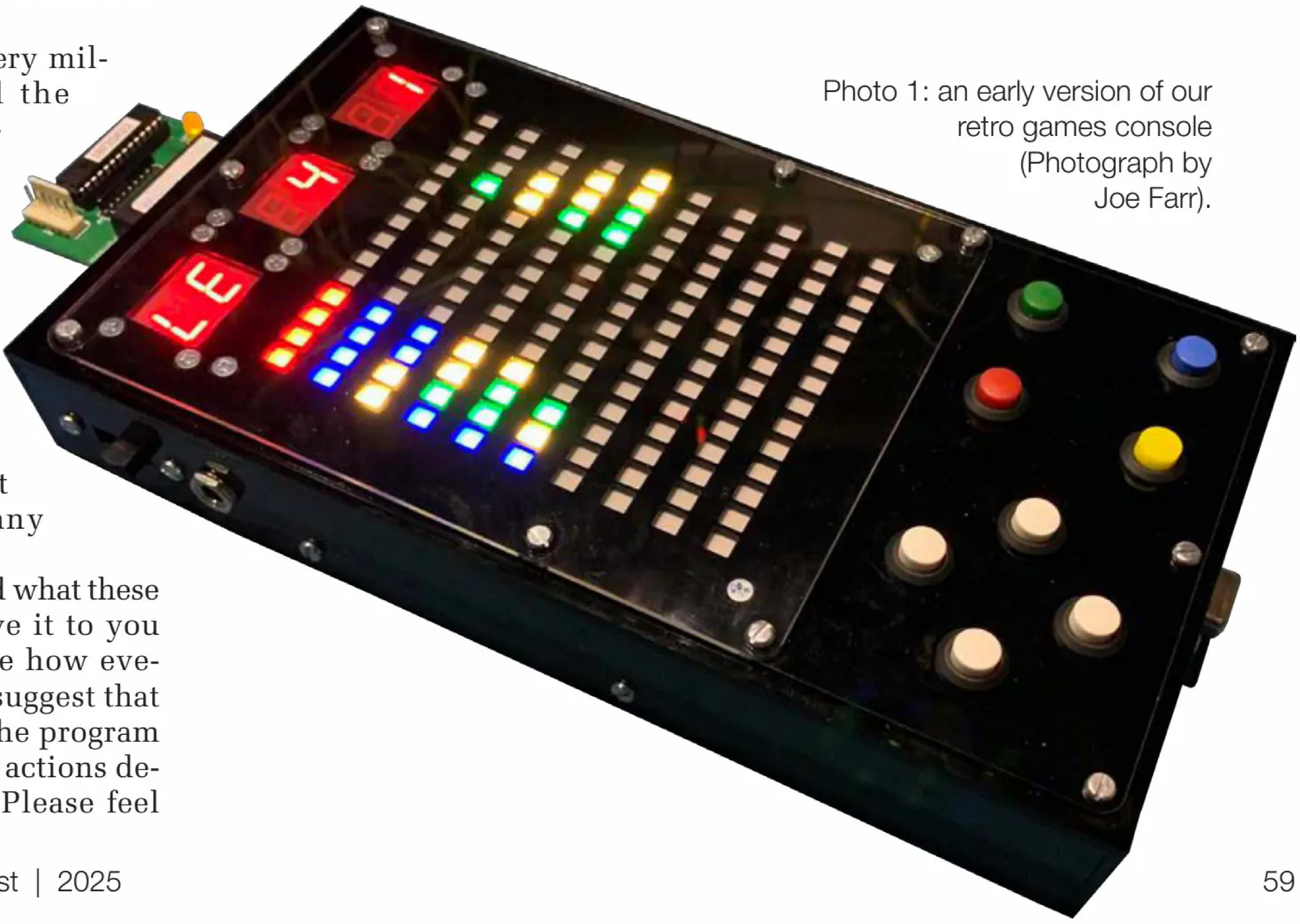


Photo 1: an early version of our retro games console (Photograph by Joe Farr).

The **Fox** Report

Barry Fox's technology column

The UK's TV Future: Linear, Digital or Just Confused?

This year's DTG (Digital Television Group) Summit was billed as "TV: The Bigger Picture 2025, a pivotal gathering at a defining moment for our industry."

To talk about the future of TV, we first need to learn a whole new language. "Generation Z" includes anyone born between 1997 and 2012. Gen Z grew up accessing the internet and using smartphones.

They are used to being able to access content of all kinds – information, music, podcasts, news and opinion, how-to-do-things video clips and commercial movies – from the Internet for near-instant access.

A few years ago, the annual conferences held by the UK's Digital Television Group (DTG) were about new developments in live over-the-air TV, bigger and flatter screen displays, time-shifting with recorders and all the electronics nuts and bolts that made it all possible.

In contrast, newer conferences are mostly about new-wave 'digital broadcasting'. This is mainly about ways of grabbing viewers' eyeballs for more than a few seconds and earning revenue from successful grabbing.

Channel 4, the UK's commercially funded public station, recently switched bosses. There are no surprises then when newly appointed chief commercial officer (CCO) Rak Patel came from Spotify with the remit "to accelerate the network's commercial growth" and manage a "seamless transition from linear to digital while maximising advertising revenue".

Speaking at the DTG event, Patel said that C4 aims to become the first 'digital' public service provider, and by 2030, will be earning 50% of its revenue from digital.

Because C4 is a public service broadcaster, it will be answerable to Ofcom (on issues such as honesty, accuracy and decency), which Internet streaming services are not. The same strictures should also affect Channel 5 broadcasting and My5 streaming, which have now relaunched as "5", a single service that unites the linear and streaming/digital platforms.

Patel was vague on what actual changes C4 would make to achieve his goals, but within days of the DTG event, Channel 4 had announced that the broadcaster's video content would be made available on Spotify. Presumably, the deal had not been signed before Patel spoke.

Arguably, the most important technical development and the enabling lynch pin to the fusion of linear (traditional) and digital 'broadcasting' is the service and system called Freely. The company behind Freely is Everyone TV (formerly Digital UK), which is backed by all the UK's public service broadcasters: the BBC, ITV, Channel 4 and Channel 5.

Freely describes itself as "the newest and most exciting way to do free TV", but was present only in the conference foyer to celebrate its first birthday with balloons and cup-cakes.

If ever there were a vitally important new development that cried out for a clear message, it is Freely; but in stark contrast to the wonderful job done by Digital UK in handling the big switch between analog and digital TV, Everyone TV just isn't messaging clearly.

The company's publicity remains a confusion of marketing hyperbole, with no clear and simple explanation of what Freely actually does and how it does it. Freely's Front of House PR seems out of their depth on anything technical, with answers to basic questions reading as if they come from a sanitised hymn sheet. So facts on Freely are surprisingly hard to come by.

The TV set-makers signed up with Freely are wary of talking to the press. This is very different from the way Digital TV operated. There must be skilled people inside Everyone/Freely who could talk clearly, but they are well hidden and probably have no idea what it is like for the outside world to show serious interest in the Freely solution.

The irony is that the problem that Freely solves is self-evident to anyone with an existing Smart TV. For example, TV station Talk TV moved from Freeview broadcast to Internet distribution but still has a Freeview channel slot (280).

A Freeview Smart TV finds Talk TV as a TV channel and then has to switch itself from digital terrestrial reception to online app access. This can take painful minutes of dark screens, loading messages and pre-recorded video clips and audio jingles, and sometimes an unexplained failure to connect.

What emerges from the Freely PR fog is that Freely uses a new program listing system which seamlessly combines off-air broadcast stations with online station sites. A new generation of Freely capable TV sets uses the Freely listings to let viewers surf for programs regardless of whether they arrive over the air or down a line.

As far as I can understand from Freely's publicity mush, a Freely TV can perform its magic when receiving programs from a combination of Freeview digital terrestrial from an aerial and the Internet, or from the Internet alone. The Internet connection can be either by Wi-Fi or Ethernet. But Freely does not work with Freesat satellite TV.

I asked Freely to confirm or correct this simple factual understanding. What I got back was a dollop of marketing speak:

All you need is Wi-Fi for a Freely TV to be able to access all your favourite shows live and on demand, all in one place. No need for a dish or aerial.

Those who choose to connect their TV via a hybrid Wi-Fi and DTT connection will also receive additional live Freeview channels delivered over DTT, alongside all the benefits of the Wi-Fi connection seamlessly integrating live and on demand shows.

Freely is not a satellite product. Everyone TV offers a TV platform for every household however they choose to connect with Freely (Wi-Fi), Freeview (DTT), and Freesat (satellite).

This unclear response even manages to confuse and contradict on the simple matter of satellite reception.

From my own hands-on experience, I can confirm that self-switching from off-air reception to online stream does not work on an LG Smart TV with Freesat free-to-air satellite reception. So, it's highly likely that Freely does not work with Freesat.

The Fox Report

Barry Fox's technology column

This matters because, in some parts of the UK (notably West Sussex), interference from foreign stations that are using the same or similar terrestrial frequencies to the UK transmitters completely swamps Freeview when the weather is warm and good for long-distance reception. Freesat is then the only free way to receive off-air 'linear' TV.

Freesat has already dropped its previously very valuable feature of letting viewers use a phone app to program a Freesat recorder remotely. For instance, to record a TV program while sitting in the pub.

The writing on the wall tells me that Freesat is on the way out. Much as the wall-writing tells that we're unlikely to get any clear answers from the 'front of house' PR people at Freely.

This was borne out at the DTG foyer display, where Freely was promising useful new features such as:

- **Backwards TV Guide**, where the viewer can easily scroll back to a program broadcast in the last week and either immediately start streaming it or add it to My List.
- **Never Miss**, which highlights the most popular shows which are either just missed or just coming up.
- **My List**, the option to save links to up to 50 personal favourite shows

All this sounds good for viewers. But viewers can only benefit by junking their existing TV set and buying a new one with Freely hardware and software locked inside. The obvious way around this is to offer a dongle like the Amazon Fire Stick, which connects to an existing TV by HDMI port and accesses the Internet by Wi-Fi.

At last year's DTG event, Freely representatives were talking about dongles in the future, but at this year's event, they seemed to have forgotten this.

"What's a dongle?", replied one of Freely's FOH people advisers on the Freely display when I asked what progress had been made on letting viewers up-date old TVs. His colleague then came to the rescue, thinking that a dongle was "something that connects with USB". (No, in this case, it connects with HDMI). The cotton-wool reply that Freely gave me when I followed through after the event is that:

Freely is currently focused on partnering with smart TVs, and in its first year has partnered with half of the UK TV market. Everyone TV, with Freeview and Freesat, has experience of launching devices such as set-top boxes. For Freely, we are continually exploring new features, new partners and availability, all part of our roadmap to continue to grow and enhance the viewer experience. Freely have not announced anything on devices outside of smart TVs.

The absence of a Freely capable Amazon Stick can't be because of technical issues; Freely is already built into Amazon Fire TVs. Reading between the lines of the official Freely hymn sheet, it's likely that the TV manufacturers see Freely as a great new way to sell new TV sets to people who are perfectly happy with their existing and often near-new sets.

It suits Freely to help them by doing no dongle deals – much as the IT industry sees Microsoft's policy of obsoleting whole rafts of perfectly good Windows 10 PCs as a great new way to sell new computers. Of course, this is terrible for consumers.

But significantly, just days after the DTH event, Tim Davie, the BBC's Director General, gave a speech to the Royal Television Society that clearly endorsed the idea of bolt-on devices such as dongles that would add Freely to existing TVs. It would surely be in the interests of all the broadcasters backing Freely to kick Freely into gear on dongles.

Several speakers at the DTG Summit referred, inevitably, to AI. One area where it is seen as a way to grow audiences and increase revenue is in helping viewers cut through the jungle of different programs on offer through an ever-increasing number of sites, service and channels. AI would learn more about viewers' likes and dislikes than they do themselves.

This is unlikely to upset anyone, because it is what Tivo pioneered decades ago.

But another use for AI is more contentious. This was mentioned by Fiona Campbell, Controller, Youth Audience at BBC iPlayer and BBC Three, but only in passing. Experiments are under way on training AI to 'watch' a football match and create a spoken commentary on the fly.

The DTG's conferences have always been notable for the excellence of the live subtitling provided for deaf members of the audience, often representing deaf-care bodies and services. In the past, the subtitling has been provided by court reporters, trained to type what's said by lawyers, judges and witnesses at real-time speed.

The DTG had found that automatic captioning can give frustratingly bad results. You can see this for yourself by switching on hard-of-hearing subtitles while watching a movie auto-captioned by YouTube's basic system.

The old film noir 'Framed' with Glenn Ford is a good example. Music is captioned as 'applause' and the displayed dialogue text often bears only passing resemblance to the actors' spoken words and lines of text appear in the wrong order (see **https://youtu.be/PoTInQtfEEs**). For example, we have seen these automatic subtitles convert the video game title "Doom Eternal" into "due maternal"!

The DTG has now partnered with Australian company AI-Media (**www.ai-media.tv**) to prove that AI subtitling is now a viable alternative to court reporting. The captions come up live, almost immediately and with next to no mistakes.

Said a spokeswoman for the company, "The solution we provided for the DTG event was our LEXI AI captioning. This was combined with our LEXI Viewer display".

The LEXI system is pre-trained with a 'Topic Model', which is essentially a library of technical words and specialist phrases that speakers are likely to use. The system was first used at the 2024 DTG Summit, and independent accuracy tests showed an average accuracy rate of 99.2% NER (Number, Edition error and Recognition error rate).

A rate of 100% would mean no mistakes made. I shudder to think what the NER is for YouTube movies.

Anyone in search of more detail of the system used can read a full study and case history on AI-Media's website: **https://www.ai-media.tv/wp-content/uploads/AI_Media_Case_Study_DTG-Summit_FINAL-14.06.24.pdf**

180-230V DC Moto

Power
0-230V DC
Output
SILICON CHIP
180V DC MOTOR CONTROLLER
0
10
20
30
40
50
60
70
80
100
SPEED
Reset
Run
WARNING!
This Speed Controller circuit operates directly from the 230VAC mains supply and contact with live components and exposed connections can be lethal, unless disconnected from the mains.

- Controls 180-230V DC motors rated from 1A to 10A (¼HP to 2.5HP)
- Controlled by four common op amp ICs with one opto-coupler and three linear regulators
- Zero to full speed control
- Safe startup procedure
- Emergency cut-out switch facility
- Automatic over-current switch-off
- Optional reversing switch capability
- PWM, Live and Power indicator LEDs
- Rugged diecast aluminium enclosure
- Current and back-EMF monitoring for speed regulation under load
- Initial setup adjustments can be done with a low-voltage supply

Motors rated at between 180V and 230V DC are supported; they are driven by PWM-chopped rectified mains. Motor load/speed feedback is via current and back-EMF monitoring. The circuit is based on analog techniques and is designed to be robust and easily adjustable. Most of the active devices are common types of operational amplifiers (op amps).

The motor speed is controlled by a rugged IGBT (insulated gate bipolar transistor). Most parts are through-hole types except for a few resistors and the opto-isolated IGBT driver. It all fits in a convenient diecast aluminium case.

Having already described the overall design and how the circuit works in the first article published last month, let's move on to building it.

Construction

Most of the parts are mounted on a double-sided, plated-through PCB coded 11104241 that measures 201×134mm, which fits in a 222 × 146 × 55mm diecast aluminium enclosure. The only off-board parts are the GPO socket for the motor, the speed potentiometer and the IEC mains input socket.

Warning: Mains Voltage

This Speed Controller operates directly from the 230V AC mains supply; contact with any live component is potentially lethal. Do not build it unless you are experienced working with mains voltages.

Some of the PCB tracks connecting to Q1, BR1 and CON1 on both sides of the PCB are tin-plated so they can handle more current. Before installing any parts, they should be covered with a layer of solder to thicken the tracks and further reduce their resistance (shown in red in Figs.3 & 4). However, be careful to avoid getting solder in the component through-holes when doing so.

Fig.3 shows the layout of the top-side parts on the PCB, which you can use as a guide during assembly. Begin by fitting the surface-mounting opto-coupled Mosfet driver (IC5). You will need a soldering iron with a fine or medium tip, a magnifier and good lighting. It's also a good idea to have a syringe of flux paste and some solder-wicking braid on hand.

Solder IC5 to the PCB pads by first placing it with its pin 1 locating dot to the top left and the IC leads aligned with the pads. Solder a corner pin and check that the IC is still aligned correctly. Soldering the small leads will be easier if you apply a small amount of flux paste on top first.

If it needs to be realigned, remelt the solder and gently nudge the IC into alignment. When correct, solder all the IC pins. Any solder that runs between and bridges them can be removed with solder wick.

Following that, mount diodes D2-D10 and zener diodes ZD1-ZD3. Ensure each is orientated correctly and that the correct diode or zener diode is installed in each location before soldering their leads. Diodes D2 and D6-D8 are 1N4004 types, while the remainder (except for the zeners) are 1N4148 types. ZD3 has a different voltage rating from ZD1 and ZD2, so don't get them mixed up.

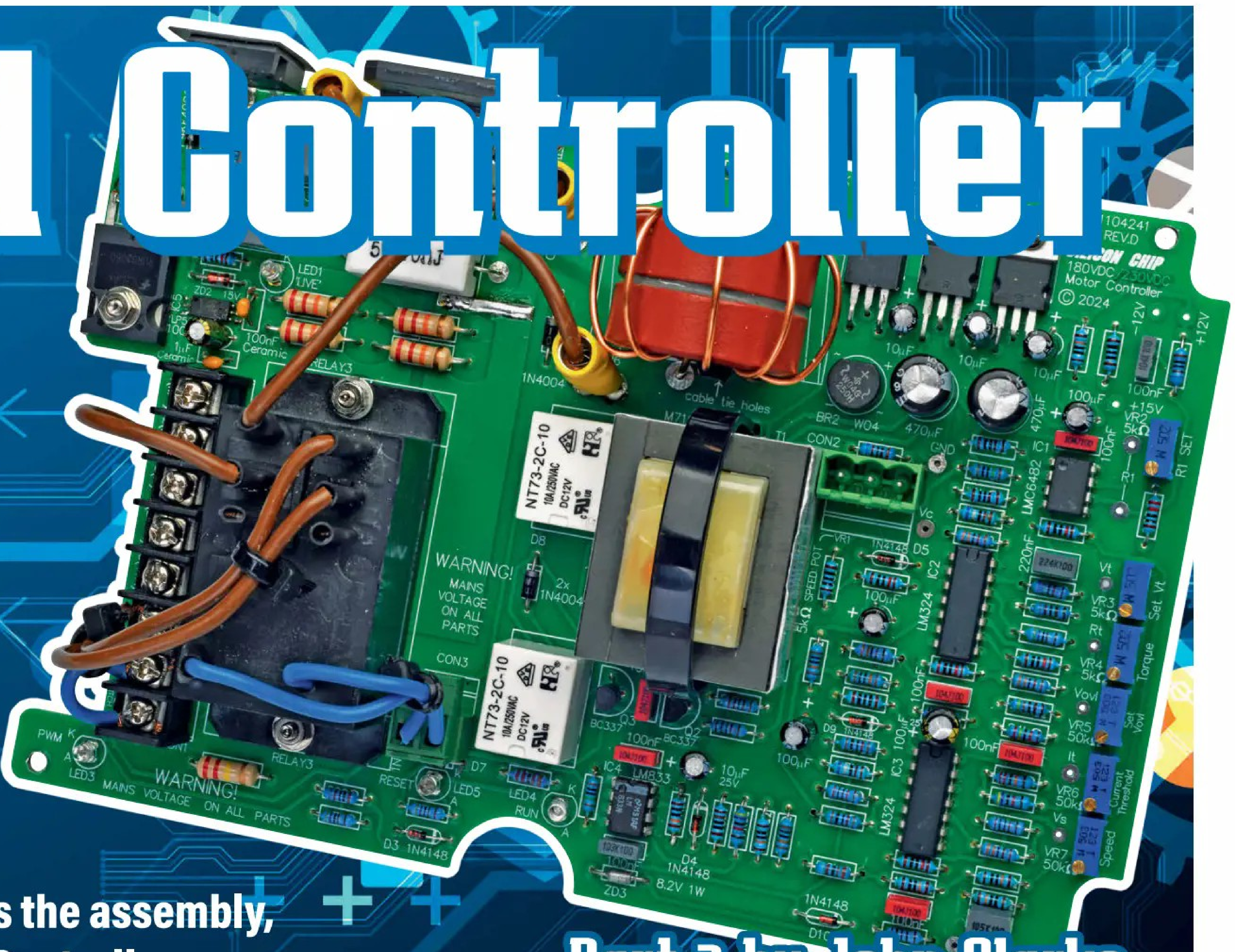
TVS1 can also be installed now; it is bi-directional, so it can go in either way around.

Follow by soldering the SMD resistors in place. These mount on the underside of the PCB, as shown in Fig.4. They are the 10kΩ resistor under IC5 and the four 0.022Ω resistors under inductor L1. If you are building the speed controller for a motor rated at 5A or less, see Table 1 for the required number and value of these shunt resistors. Otherwise, fit all four.

Install these by soldering one end first, then the other after you've checked that the part is aligned correctly and the first solder joint has solidified.

The low-wattage (½W) through-hole axial resistors can be fitted now. They

r Speed Controller



Our new High-voltage DC Motor Speed Controller, revealed last month, can control motors commonly used in lathes and treadmills. It can operate such a motor from stopped up to full speed and maintains a constant speed even with a varying load. This article covers the assembly, testing and setup of the Speed Controller.

Part 2 by John Clarke

have colour-coded bands and the codes were shown in the parts list last month. However, you should also use a digital multimeter to check each resistor before soldering it, as the colour bands can be difficult to distinguish (especially brown, red and orange).

The remaining ICs can now be installed, taking care to get the correct one in each place and with pin 1 in the proper location (double check that!). Sockets can be used for each of the ICs, although they can also be soldered directly to the PCB, which is likely to give better long-term reliability.

The 1W and 5W resistors can be mounted next. When fitting the 5W resistors, leave a gap of around 1mm between the body and the PCB to allow air to circulate.

Regulators

REG1-REG3 mount horizontally with their leads bent by 90° to fit into the allocated holes in the PCB. Each regulator is secured to the board using a 6mm-long M3 screw and nut before the leads are soldered. Make sure you don't mix up the three different regulator types.

Diode D1 is mounted horizontally with its leads bent by 90° so you can insert them into the PCB holes. Secure it with an M3 screw, washer and nut before soldering.

The capacitors can now be installed. There are four types used. The 630V-rated 47nF capacitors operate at rectified mains voltage, so make sure the correct types are used in the upper-left corner of the PCB. The other types are ceramic, MKT polyester and electrolytic. The 100nF and 1µF ceramic capacitors are placed near IC5.

The electrolytic capacitors need to be orientated correctly since they are polarised. The longer lead indicates the positive side, while the negative stripe down one side of the capacitor indicates the negative side. One 100µF (just above IC3) and 10uF capacitor (above D4) is rated at 25V, so ensure it is located correctly, or it will be damaged when power is applied.

For the MKT and ceramic capacitors, the 10nF capacitor is likely to be marked 103, the 100nF capacitors marked 104, the 220nF capacitor marked 224, and the 1µF capacitor marked 105.

Solder in the three PCB-mounting spade connectors (CON5-CON7), then bridge rectifiers BR1 and BR2, ensuring correct orientation. BR1's positive lead is spaced further from the others, so it will only fit in one way. Mount it so there is about 1mm of lead length below the PCB for soldering. For BR2, the longer lead is positive. The AC and + terminals will also be marked on the package.

LED1-LED5 can be fitted now. Be sure they are correctly orientated with the longer lead placed into the anode (A) hole in each case. The power LED,

Table 1 – shunt resistor values depending on motor rating

HP	¼	½	¾	1	1¼	1½	1¾	2	2¼	2½
kW	0.18	0.36	0.54	0.72	0.9	1.08	1.26	1.44	1.62	1.8
A	1	2	3	4	5	6	7	8	9	10
0.022Ω shunts	2S*	2S*	3	3	3	4	4	4	4	4
VR2 value (R1)	4.95kΩ	2.25kΩ	1.95kΩ	1.36kΩ	1.47kΩ	1.36kΩ	1.1kΩ	913Ω	764Ω	645Ω
IC1b gain	12.5	6.25	5.55	4.16	3.33	4.16	3.57	3.125	2.77	2.5

** S = in series. Alternatively, you can use two 0.05Ω resistors in series and set R1 to 1.93kΩ (¼HP) for a gain of 5.5 or 752Ω (½HP) for a gain of 2.75*

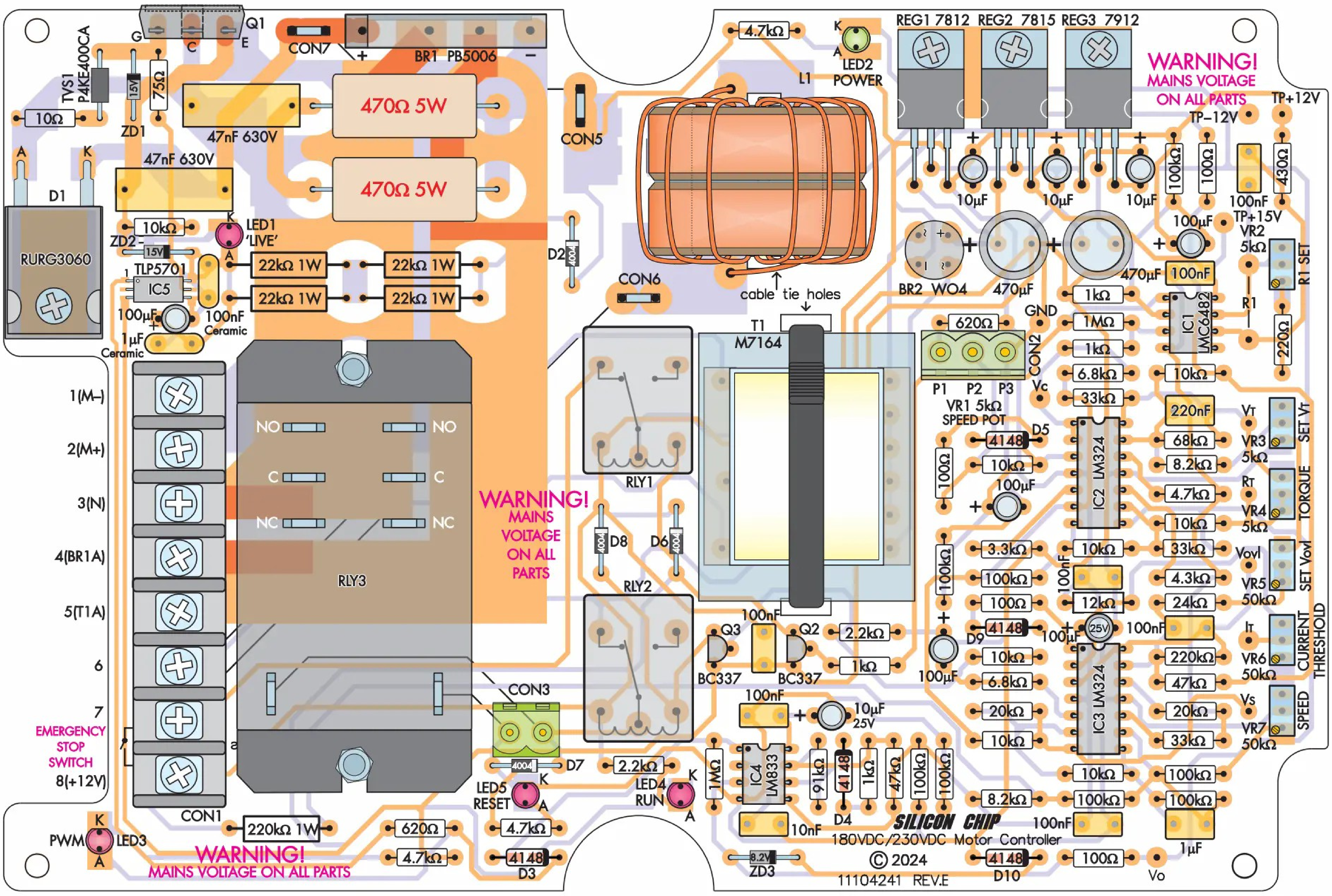

Q1
G C E
CON7
BR1 PB5006
TVS1 P4KE400CA
15V
75Ω
ZD1
10Ω
470Ω 5W
470Ω 5W
47nF 630V
47nF 630V
CON5
A K
D1
RURG3060
10kΩ
ZD2
LED1 'LIVE'
TLP5701
IC5
22kΩ 1W
22kΩ 1W
22kΩ 1W
22kΩ 1W
D2
100μF
100nF Ceramic
1μF Ceramic
CON6
4.7kΩ
LED2 POWER
L1
REG1 7812 REG2 7815 REG3 7912
WARNING! MAINS VOLTAGE ON ALL PARTS
TP+12V
TP–12V
100kΩ
100Ω
100nF
430Ω
10μF
10μF
10μF
100μF
TP+15V
VR2 5kΩ
R1 SET
BR2 W04
470μF
470μF
100nF
R1
cable tie holes
T1 M7164
GND
620Ω
1kΩ
1MΩ
1kΩ
CON2
IC1 MC6482
220Ω
P1 P2 P3
Vc
6.8kΩ
10kΩ
VR1 5kΩ SPEED POT
33kΩ
220nF
Vt
D5
4148
68kΩ
VR3 5kΩ
SET Vt
100Ω
10kΩ
IC2 LM324
8.2kΩ
100μF
Rt
4.7kΩ
TORQUE
VR4 5kΩ
10kΩ
1(M–)
2(M+)
3(N)
4(BR1A)
5(T1A)
6
7
EMERGENCY STOP SWITCH
8(+12V)
NO NO
C C
NC NC
RLY3
RLY1
WARNING! MAINS VOLTAGE ON ALL PARTS
D8 D6
RLY2
3.3kΩ
10kΩ
33kΩ
Vovl
SET Vovl
100kΩ
100kΩ
100nF
4.3kΩ
100Ω
12kΩ
24kΩ
VR5 50kΩ
100nF
Q3 Q2
2.2kΩ
4148
D9
100μF
25V
100nF
It
CURRENT THRESHOLD
BC337 BC337
1kΩ
10kΩ
100μF
220kΩ
VR6 50kΩ
6.8kΩ
47kΩ
IC3 LM324
CON3
100nF
10μF 25V
20kΩ
20kΩ
Vs
SPEED
VR7 50kΩ
10kΩ
33kΩ
D7
2.2kΩ
1MΩ
IC4 LM833
91kΩ
4148
1kΩ
47kΩ
100kΩ
100kΩ
LED5 RESET
LED4 RUN
10kΩ
100kΩ
CON1
8.2kΩ
100kΩ
100kΩ
100kΩ
D4
SILICON CHIP
100nF
220kΩ 1W
620Ω
4.7kΩ
10nF
180VDC/230VDC Motor Controller
1μF
PWM
LED3
WARNING! MAINS VOLTAGE ON ALL PARTS
4.7kΩ
4148
D3
8.2V
ZD3
© 2024
4148
100Ω
11104241 REV.E
D10
Vo

Fig.3: most components mount on the top side of the PCB. T1 and L1 are heavy, so both are secured to the PCB using cable ties. The large relay, RLY3, is attached to the board using screws and nuts, and will later be wired to CON1 and CON3. This diagram and Fig.4 are both shown at 90% of actual size. The red areas are where extra solder is added.

LED2, is green while the remainder are red.

LED1 and LED3 can be mounted vertically with 5mm of the leads projecting above the top PCB surface. LED2, LED4 and LED5 display the Power, Reset and Run status on the front lid via fibre-optic light transporters. Before soldering each LED, clip the LED bezel end piece for the fibre optic connection onto the LED, then solder it in place with the clip touching the PCB surface.

CON1 to CON3 can be fitted now. CON1 can be installed either way around, but CON2 and CON3 must be orientated correctly. That is most easily done by plugging the screw terminal plug into each socket before mounting it to the PCB. For the 3-way terminal, CON2, the wire entry faces toward diode D5. For the 2-way terminal, CON3, the wire entry faces away from diodes D7 and D3.

The next step is to install the relays. RLY1 and RLY2 directly mount onto the PCB, while RLY3 is held to it using M3 screws, washers and nuts, with each screw inserted from the underside of the PCB. The washers go under the nuts on top of the relay's mounting feet.

T1 can now be mounted onto the PCB. Its pins hold it in place, but we use a large cable tie to ensure it cannot move and break the transformer pins if it is dropped. There are slots in the PCB to accommodate the cable tie, to wrap around the transformer body and under the PCB after soldering it in place.

Winding inductor L1

L1 is made using two powdered iron cores side-by-side. Use epoxy resin to glue the two cores together. Once the glue has set, wind on seven turns of 1.25mm-diameter enamelled copper wire. The winding direction is not important.

The finished winding and core are mounted on the PCB with a cable tie securing the toroid to the PCB. This tie is fed through the slots in the PCB to wrap around through the centre hole of the core and under the PCB.

Trim the wires to sufficient lengths to solder to the PCB pads, then strip the insulation off the ends of the enamelled wire. It's generally best to do that with a sharp hobby knife (be careful not to cut your fingers!) or emery paper. Depending on the enamel used, you may be able to burn it off by holding a blob of molten solder over the wire ends.

Make sure the enamel is entirely removed so you can make a good solder joint, then solder the wire ends to the pads for L1.

Q1 can be installed now. Stand it above the PCB so there is about 1mm of lead projecting below the PCB to allow soldering. Because the PCB tracks near the IGBT are thin, the exposed, tinned PCB tracks at the emitter and collector should be built up with solder to lower the resistance.

Final assembly

The PCB is secured inside the enclosure base using M3.5 screws into the integral standoffs in the base. However, before attaching the PCB, the IEC con-

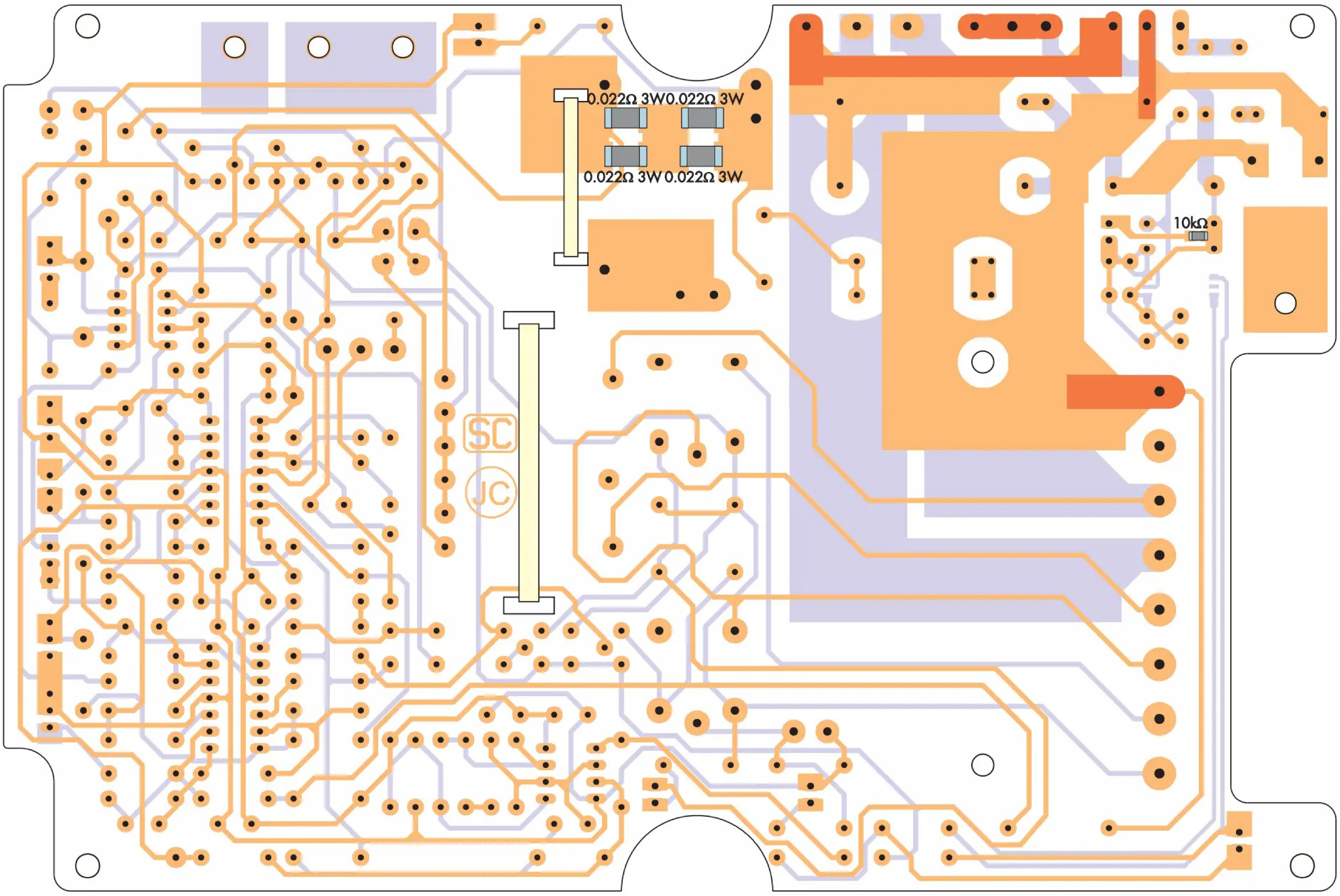

0.022Ω 3W
0.022Ω 3W
0.022Ω 3W
0.022Ω 3W
10kΩ
SC
JC

Fig.4: if all four 0.022Ω shunt resistors are soldered to the board, as shown here, the Controller will suit motors rated at 6-10A (1.5HP+, 1.08kW+). For lower-powered motors, fewer resistors are fitted, as per Table 1. For ¼HP and ½HP motors, make sure the two resistors are in series, not parallel. If three resistors are required, any three can be fitted.

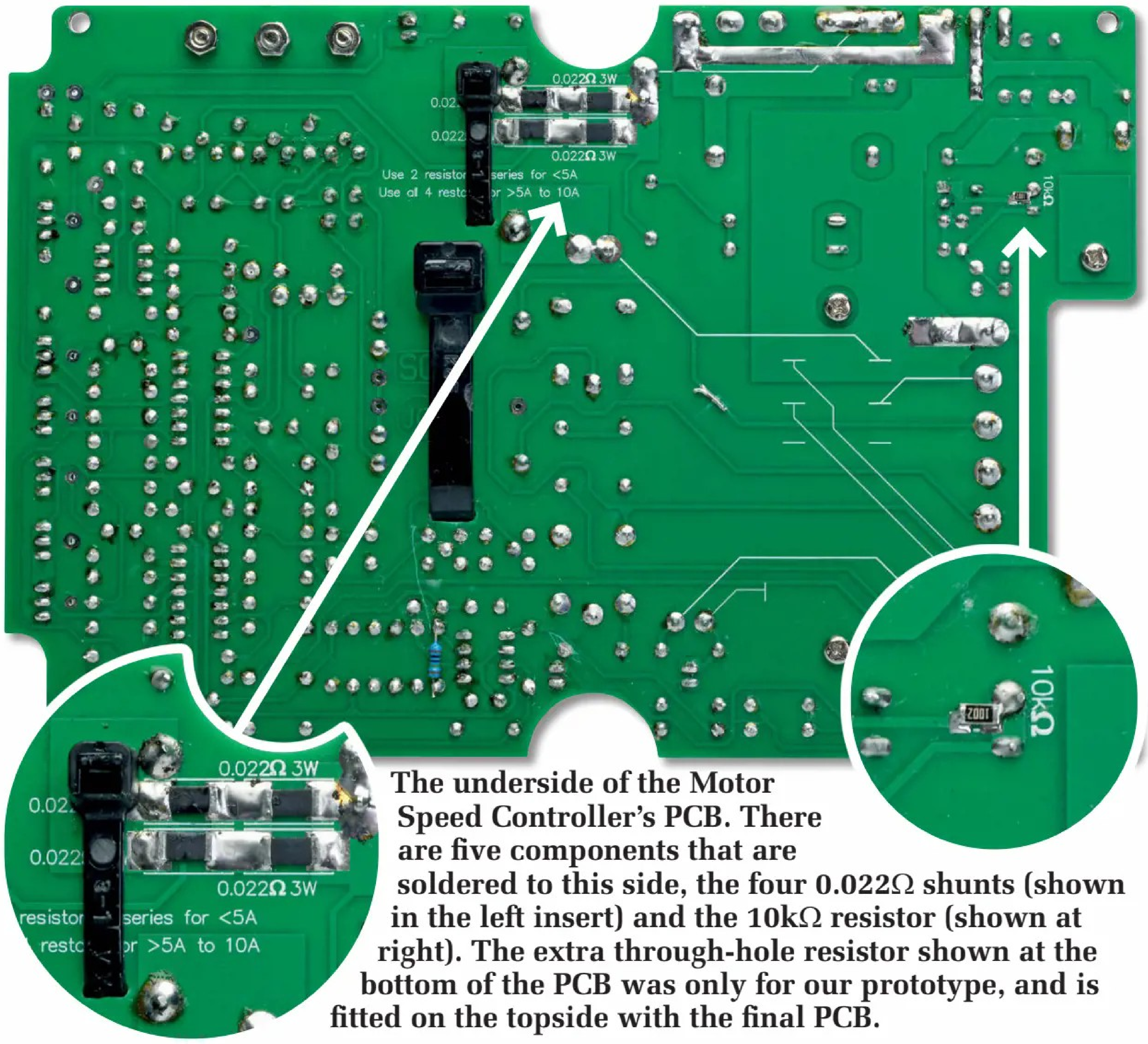
nector cutout will need to be made in the side of the enclosure.

You will need to drill and shape holes in one end of the case for the IEC connector and Earthing screw. You might as well prepare the lid at the same time, which needs holes made for the GPO socket, Earthing screw and speed control potentiometer.

Fig.5 is a guide for the required cutouts; it can be copied or downloaded and used as a template. The large cutouts for the mains GPO and IEC connector can be made by drilling a series of small holes around the inside perimeter, then knocking out the centre piece and filing the job to a smooth finish.

Alternatively, you can use a speed bore drill to remove a large portion of the required area and then file it to the final shape.

The Earth screw positions are not critical. Use the wiring diagram (Fig.6) to decide where to place the holes. One 4mm hole is required on the lid, and one in the enclosure base. Two 3mm holes are needed to secure the IGBT (Q1) and BR1 against the side of the


0.022Ω 3W
0.022Ω 3W
Use 2 resistors in series for <5A
Use all 4 resistors for >5A to 10A
10kΩ
0.022Ω 3W
0.022Ω 3W
resistor series for <5A
resto or >5A to 10A
10kΩ

The underside of the Motor Speed Controller's PCB. There are five components that are soldered to this side, the four 0.022Ω shunts (shown in the left insert) and the 10kΩ resistor (shown at right). The extra through-hole resistor shown at the bottom of the PCB was only for our prototype, and is fitted on the topside with the final PCB.

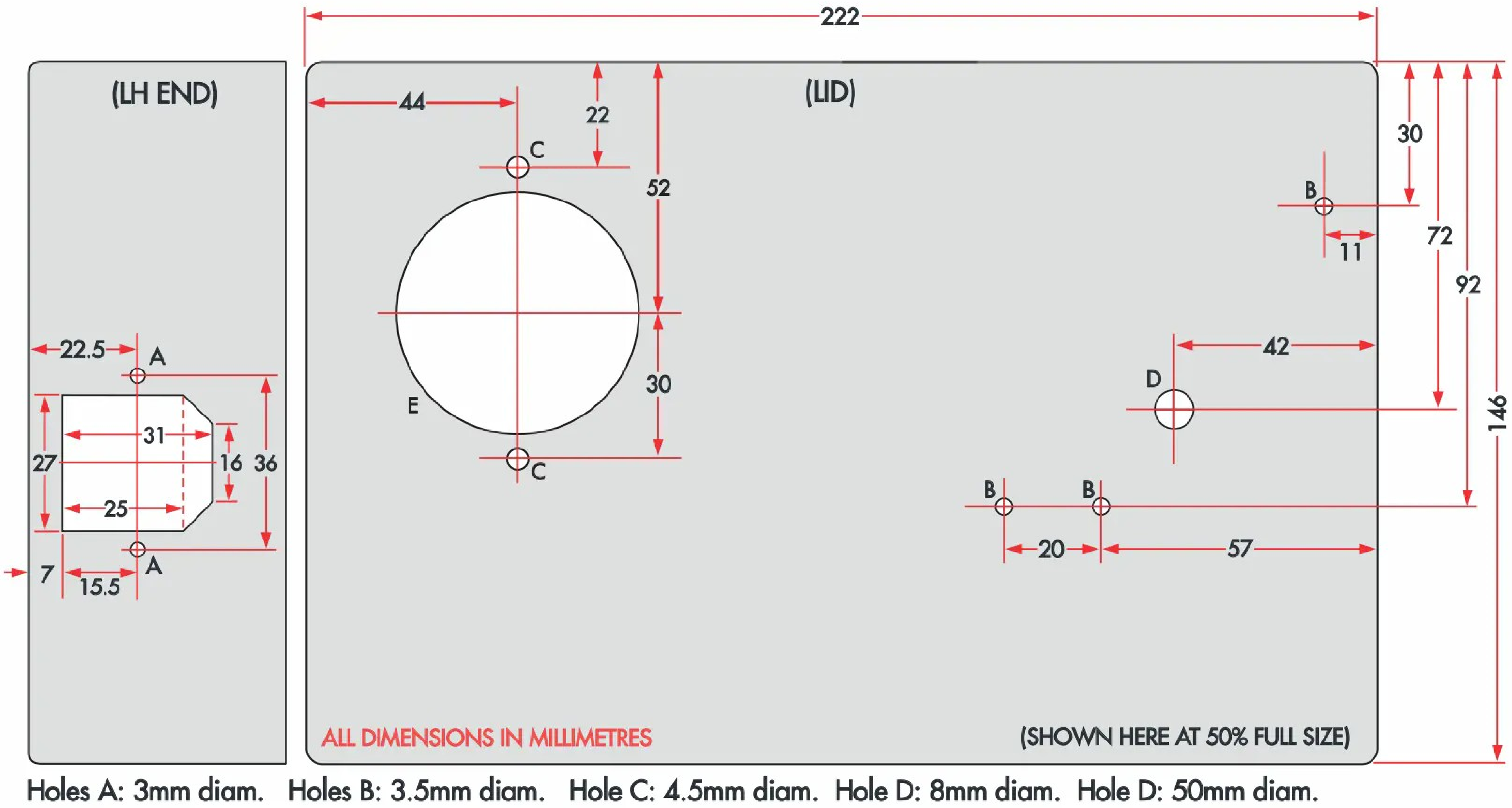

(LH END)
(LID)
222
44
22
52
30
C
C
E
D
B
B
B
30
72
92
146
11
42
20
57
22.5
A
A
31
27
25
16
36
7
15.5
ALL DIMENSIONS IN MILLIMETRES
(SHOWN HERE AT 50% FULL SIZE)
Holes A: 3mm diam. Holes B: 3.5mm diam. Hole C: 4.5mm diam. Hole D: 8mm diam. Hole D: 50mm diam.

Fig.5: the required holes in the lid and base of the diecast aluminium case. Ensure the IEC and GPO socket holes are shaped correctly (filed carefully to size) so they are not loose. The exact Earth screw positions are not critical, so they are not marked.

enclosure. Temporarily place the PCB into the enclosure and mark where the holes for Q1 and BR1 are needed.

The holes for Q1 and BR1 need to be slightly countersunk on the inside of the enclosure to provide a flat mounting surface. There must not be any sharp edges around the hole or any remaining swarf that could puncture the silicone insulating washer.

3.5mm diameter holes are needed for the fibre-optic LED bezels on the front panel. These are not directly above the LED position on the PCB to give the fibre optic cables room to flex in an 'S' shape when the lid is attached.

Once the drilling and filing is complete, install the IEC connector using countersunk head 10mm-long M3 machine screws and nuts. The PCB can then be placed inside the case but wait to secure it to the corner posts.

Q1 needs to be insulated from the enclosure using a TO-247-sized silicone insulating washer. Its package has an insulated hole, so no insulating bush is required to insulate the package from the screw. A 12mm-long M3 screw and M3 nut can be used to secure Q1 to the side of the enclosure.

Check that the enclosure is insulated from all three of Q1's leads by measuring the resistance between each lead and the enclosure. There should be high resistance reading in each case, in the megohms region.

BR1 does not require an insulating washer since the metal tab on the back of the package is insulated from the internal diodes.

Before attaching the mains GPO, you can print out the front panel label (Fig.7), available to download from our website at https://pemag.au/Shop/11/436 Details on making a front panel are found at: https://pemag.au/Help/FrontPanels

Now wire everything up per Fig.6. All wiring must be run using mains-rated cable. Be sure to use 10A cable where indicated, and note that brown wire is used for the Live wiring and blue for Neutral. Green/yellow-striped wire must be used for the Earth wiring only, and the Earth lead from the IEC connector is attached via a crimp eyelet to the enclosure Earth.

The wiring not marked as 10A can be lighter-duty 7.5A mains wire, such as for the speed potentiometer VR1, or use 10A wiring throughout.

The IEC socket and Earth screw are on the left-hand side of the case.

The terminals on relay RLY3 will be too tall for the lid to fit, so they need to be cut down, with the wires soldered directly to the shortened terminals and covered in heatshrink insulation. These terminals are brittle, so hold the lower part of the terminal with small flat-nosed pliers while you break off the top part with another set of pliers. The terminals will break at the wire hole location.

Be sure to insulate all the connections with heatshrink tubing for safety, and cable tie the wires to prevent any wire breakages coming adrift, as shown in Fig.6. The Live and Neutral leads are secured to the GPO using cable ties that pass through the holes in its moulding.

Use neutral-cure silicone sealant (eg, Roof and Gutter silicone) to cover the Live bus piece that connects the Live pin to the fuse at the rear of the IEC connector.

Take great care when making the connections to the mains socket (GPO). In particular, be sure to run the leads to their correct terminals; the GPO has the L, N and E clearly labelled. Do the screws up tightly so that the leads are held securely. Similarly, make sure that the leads to CON1's screw terminals are firmly secured.

Warnings

Almost all of the circuitry operates at mains potential, so it is dangerous to make contact with any part of the circuit when it is powered. The speed potentiometer connections are also at mains potential.

180-230V DC Speed Control Part 2

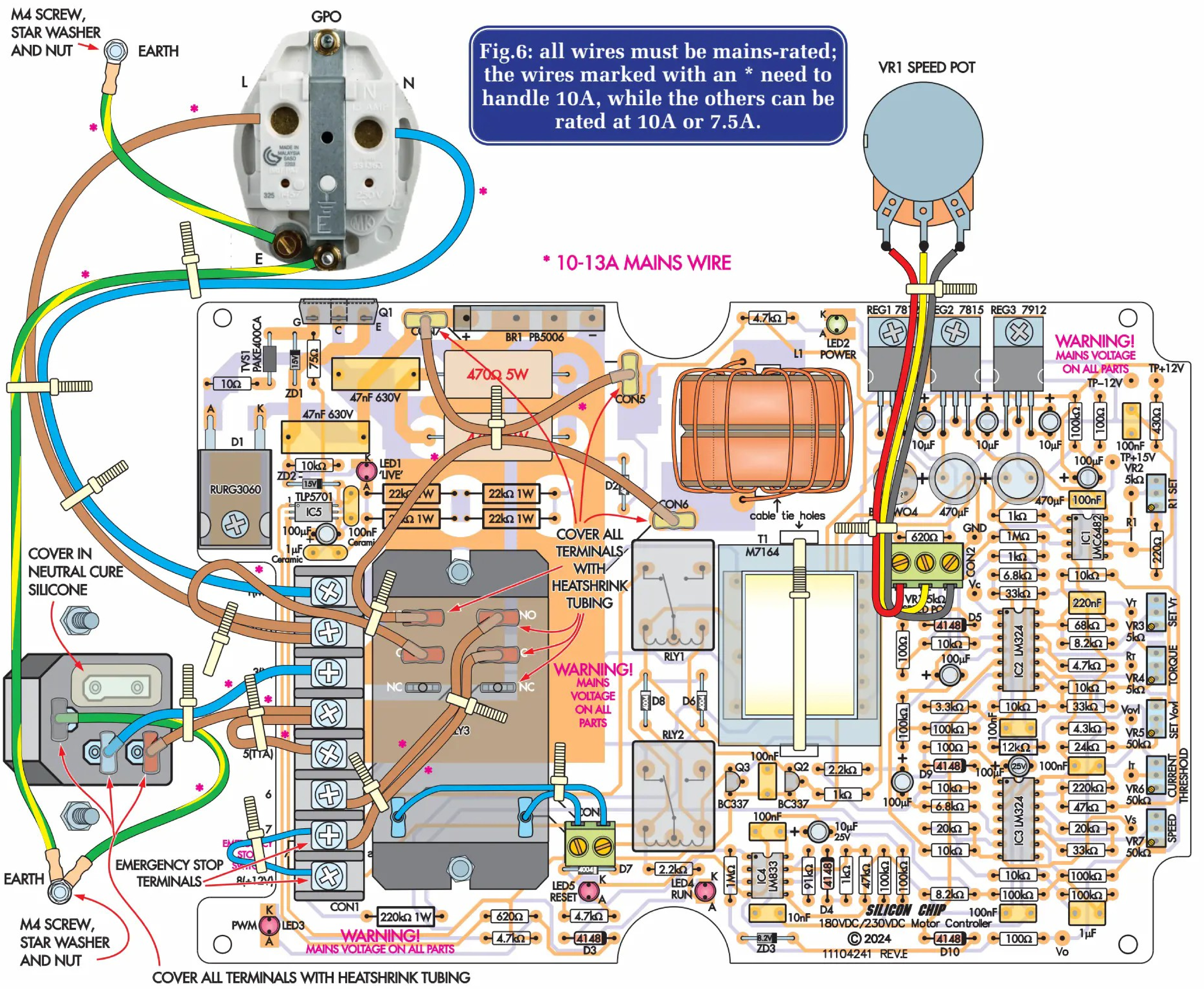


Fig.6: all wires must be mains-rated; the wires marked with an * need to handle 10A, while the others can be rated at 10A or 7.5A.

M4 SCREW, STAR WASHER AND NUT
EARTH
GPO
L
N
E
* 10-13A MAINS WIRE
VR1 SPEED POT
COVER IN NEUTRAL CURE SILICONE
EARTH
EMERGENCY STOP TERMINALS
M4 SCREW, STAR WASHER AND NUT
COVER ALL TERMINALS WITH HEATSHRINK TUBING
COVER ALL TERMINALS WITH HEATSHRINK TUBING
WARNING! MAINS VOLTAGE ON ALL PARTS
WARNING! MAINS VOLTAGE ON ALL PARTS
WARNING! MAINS VOLTAGE ON ALL PARTS
TVS1 P4KE400CA
Q1
BR1 PB5006
470Ω 5W
47nF 630V
D1
RURG3060
ZD1
ZD2
LED1 'LIVE'
TLP5701
IC5
22kΩ 1W
CON5
CON6
CON7
REG1 7815
REG2 7815
REG3 7912
LED2 POWER
L1
T1 M7164
cable tie holes
RLY1
RLY2
RLY3
D8
D6
Q2 BC337
Q3 BC337
IC1 LMC6482
IC2 LM324
IC3 LM324
IC4 LM833
VR2 5kΩ
VR3 5kΩ
VR4 5kΩ
VR5 50kΩ
VR6 50kΩ
VR7 50kΩ
R1 SET
SET Vt
TORQUE
SET Vovl
CURRENT THRESHOLD
SPEED
TP+12V
TP-12V
TP+15V
NO
NC
C
LED3 PWM
LED4 RUN
LED5 RESET
CON1
220kΩ 1W
SILICON CHIP
180VDC/230VDC Motor Controller
© 2024
11104241 REV.E

Some adjustments can be made more safely by disconnecting the mains supply from some parts of the circuit. This leaves the circuit floating at mains Neutral potential instead of Live. You still need to be very careful, but the risk of electrocution should you touch the circuitry is much lower.

Some adjustments will need to be made when the circuit is live. We recommend using a 1000V-rated screwdriver with a 0.4mm-thick, 2.5mm-wide flat tip. That size of screwdriver suits the trimpot adjustment screws and has a sufficient voltage rating to protect against electrical shocks. We used a Wiha 1000V screwdriver that has an insulated shank.

Similarly, when measuring voltages in the circuit, use a 600V CAT III (or higher, eg, CAT IV) rated multimeter and probes.

We provide indicator LEDs that show when the circuit is powered and live. So don't touch the circuit when any LED is lit, and always unplug it from the mains before working on it (except during the part of the setup where it needs to be operating).

Another LED shows the PWM duty cycle by varying brightness with PWM duty. A separate LED shows when the speed potentiometer is rotated fully anti-clockwise. Finally, there is one LED that shows when the motor can be started.

Testing

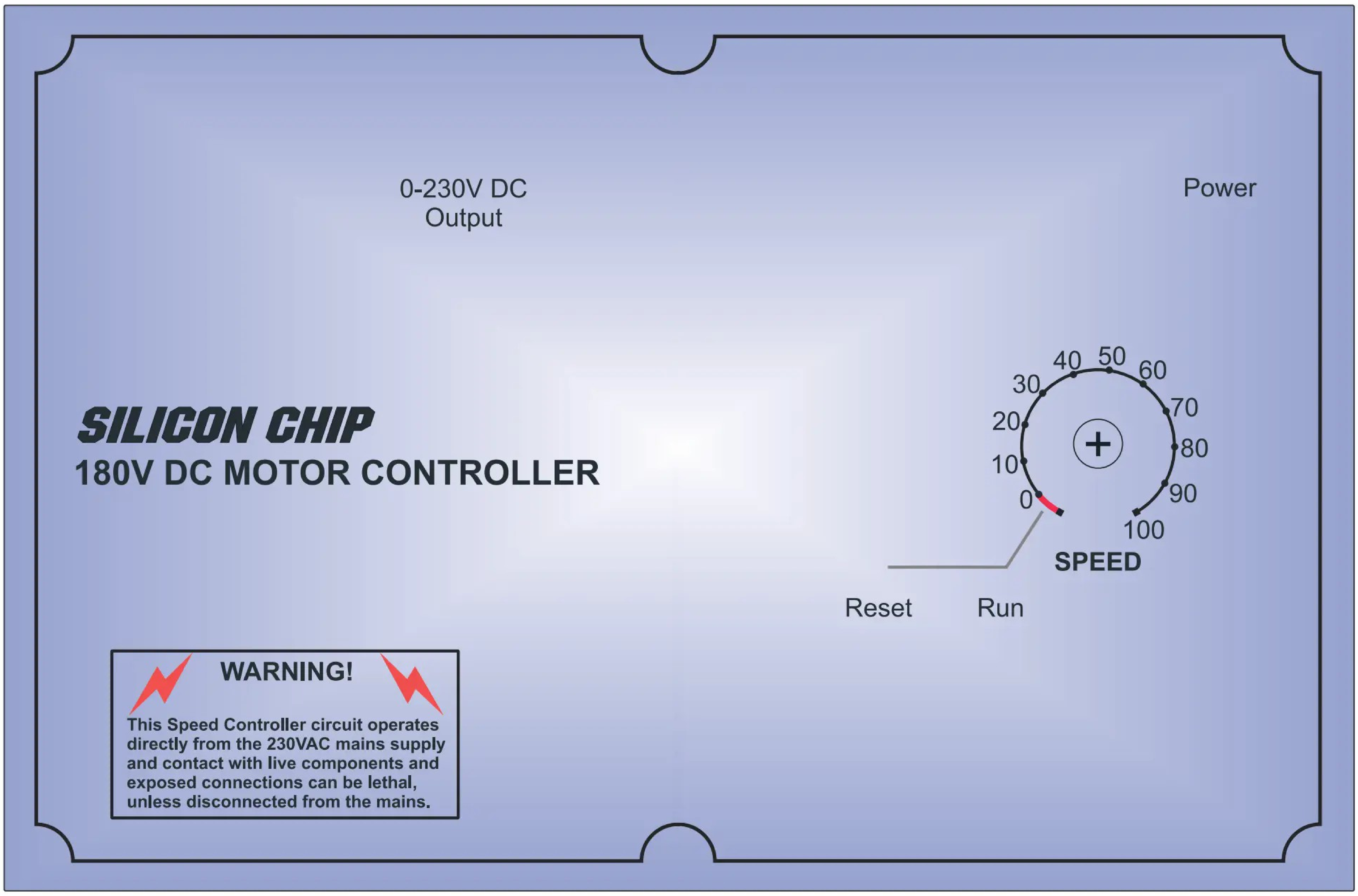
Ensure that the mains power point you connect to when testing and adjusting the 180V DC Motor Controller is connected to an Earth-leakage core balanced relay, also known as a Residual Current Device (RCD) or mains safety switch. This can be installed in the fusebox, as a separate unit within the power point or as a plug-in device.

The RCD is designed to cut the power should you receive an electric shock that passes through your body to Earth. However, an RCD will not protect you if current flows through your body from Live to Neutral (eg, by touching two points in the circuit with different hands).

Thus, it is a good idea to use one hand only when there is any possibility of making contact with the live circuitry within the Motor Controller.

If you are building the Controller in a different arrangement than the one we described, eg, with the motor hardwired to it, any wiring that goes outside the enclosure must be run in sheathed mains-rated cable that is secured to the enclosure with cordgrip grommets. This includes the safety/emergency stop switch wires.

The safety/emergency stop switch must be mains-rated and enclosed in an Earthed enclosure with its contacts covered in heatshrink tubing and the wiring cable tied together. Treat all connections to it as if they are live! Additionally, the wires from the safety switch need to be secured to its enclosure with a cordgrip grommet.



This Speed Controller circuit operates directly from the 230VAC mains supply and contact with live components and exposed connections can be lethal, unless disconnected from the mains.

Fig.7: this panel label is also available as a PDF download from the Silicon Chip website. It can be printed, laminated and attached to the lid. This panel is shown at 83.3% actual size, and so needs to be enlarged by exactly 20%.

Initial settings that can be made with the power off include setting the torque trimpot to near 0Ω. This resistance can be measured between the V_T and R_T test points.

Setting IC1b's gain

IC1b's gain is set by varying VR2's resistance (referred to as R1) to provide the required current measurement output voltage at the rated current of the motor that's used.

Table 1 shows the resistance setting for R1 versus motor ratings and shunt resistances. We show values based on ¼HP increments from ¼HP through to 2½HP. This closely corresponds to 1A to 10A motor ratings in 1A steps. That's because, for a 180V DC motor, each amp is 180W. Since 1HP is 746W, 180W is 0.24HP or near enough to 0.25HP.

Shunt values

Note that the shunt resistance comprises series and parallel resistors to provide the required overall shunt resistance. For the 1A and 2A rated motors, you can use two 0.05Ω resistors in series to form a 0.1Ω shunt instead of the 0.022Ω shunts used for other ratings; in that case, less gain is required from IC1b.

The gain for IC1b is set so that, at the full rated motor current, its output sits at 0.55V. For example, when the current shunt is 0.022Ω, and the motor is rated at 10A, the voltage across the shunt will be 0.22V at 10A. IC1b needs to amplify this to give the 550mV output, meaning a gain of 2.5. The formula for the required R1 resistance is (gain – 1) × 430Ω. That works out to 645Ω in this example.

With the power off, connect your multimeter probes to the two R1 test points on the PCB and adjust VR2 for the value required for R1. Now insert IC1 and the remaining ICs in their sockets if you have not already done so.

Overload setting

At the motor's rated current, IC1b's output delivers 0.55V. IC3d amplifies this by 4.68, giving a 2.57V output. VR6 provides adjustment of the current threshold (I_T) for motor overload. To set the motor overload to 1.6 times the rated current, the 'I_T' setting should be 1.6 × 2.57V = 4.1V. This value assumes that the V_{OVL} offset output from IC3d is set to 0V using VR5, which we will do later.

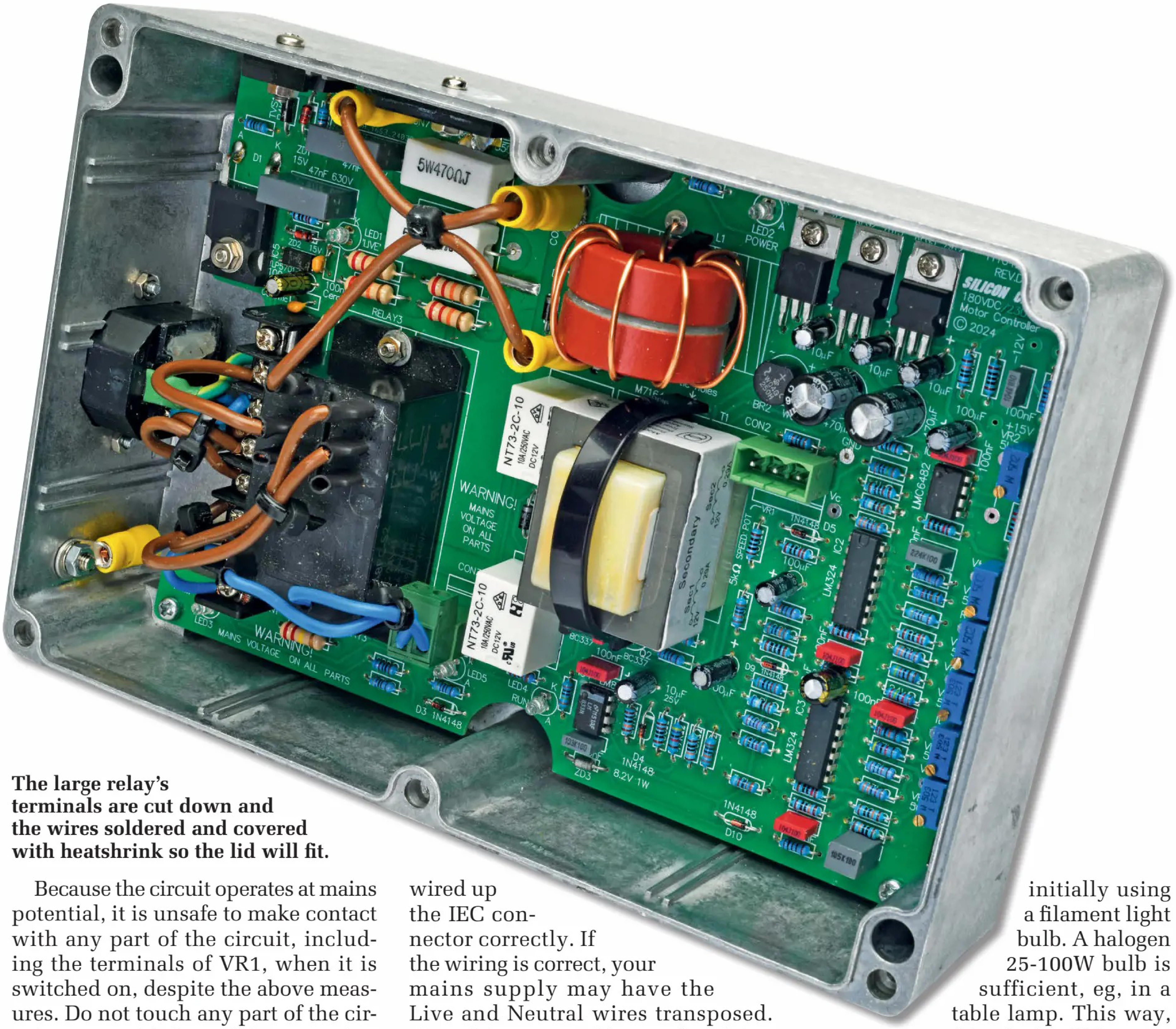
The overload trip voltage needs to be set with the power on. Before applying power:

1. Check your wiring carefully and ensure all mains connections are covered in heatshrink tubing and the wiring is cable-tied.
2. Check the Earth connection between the enclosure and the Earth pin on the IEC connector. The reading should be steady and under 1Ω.
3. Install the fuse inside the fuse holder.

Testing

The initial testing and setting up can be done more safely by disconnecting the Live wire to BR1. This is done at CON1, by removing the wire between terminals 4 and 5 and only connecting the Live wire to terminal 5. Also disconnect the spade connector wire loop between CON5 and CON7.

180-230V DC Speed Control Part 2


5W470ΩJ
NT73-2C-10
WARNING! MAINS VOLTAGE ON ALL PARTS
SILICON CHIP 180VDC Motor Controller © 2024
LM324
LMC6482

The large relay's terminals are cut down and the wires soldered and covered with heatshrink so the lid will fit.

Because the circuit operates at mains potential, it is unsafe to make contact with any part of the circuit, including the terminals of VR1, when it is switched on, despite the above measures. Do not touch any part of the circuit except with the multimeter probes and 1000V-rated screwdriver.

Attach the enclosure lid before switching on power for the first time. That will make it safer if something is wrong, such as a reverse-connected electrolytic capacitor or if a 16V capacitor is installed in a 25V position. Still, check those aspects again before fitting the lid and applying power.

If all is quiet when power is applied (except for relays clicking), switch off the power and open the lid. Wearing safety glasses, switch on the power again and measure the AC voltage between Neutral (at terminal 3 of CON1) and the mains Earth connection to the enclosure. The reading should be no more than a few volts. You should read close to 230V AC between Earth and terminal 5 of CON1.

If the Neutral reading is instead close to 230V AC, check that you have wired up the IEC connector correctly. If the wiring is correct, your mains supply may have the Live and Neutral wires transposed. Have this corrected by an electrician before proceeding with testing the motor controller.

Switching on power again, you should be greeted with power LED2 lighting to show that the +12 and -12V supplies are up. Check the regulators for the correct output voltages. There should be +12V between the 0V and +12V test points. Similarly, there should be +15V at the +15V test point and -12V at the -12V test point. These voltages should be within 5% of the designated voltages.

That means between 11.4V and 12.6V for 12V, -11.4V to -12.6V for -12V and between 14.25V and 15.75V for +15V.

Verify that when VR1 is fully anticlockwise, LED4 is off and only switches on once the speed pot (VR1) is rotated clockwise slightly.

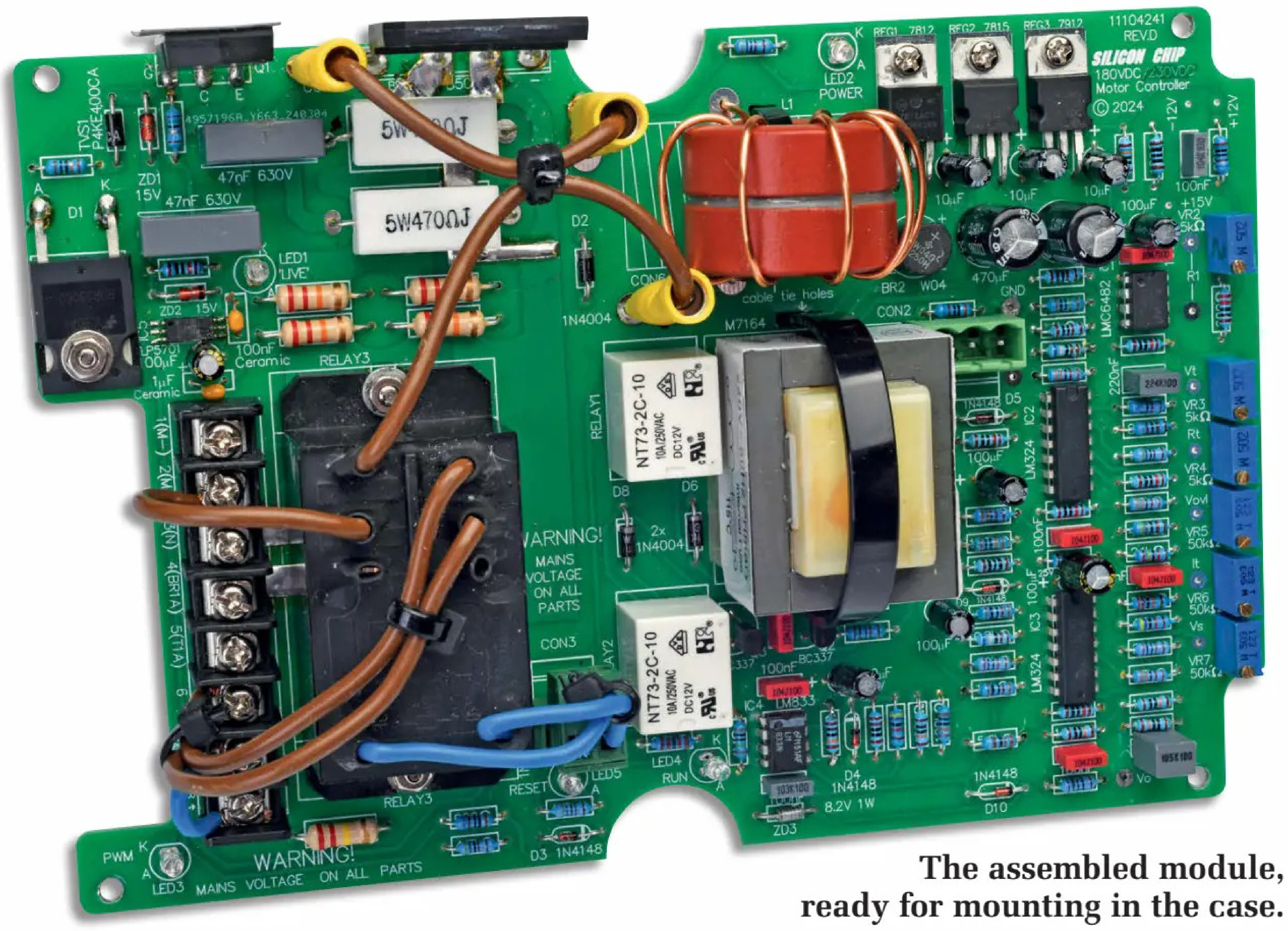
It is important to test the Controller initially using a filament light bulb. A halogen 25-100W bulb is sufficient, eg, in a table lamp. This way, nothing bad will happen if the 'motor speed' oscillates; any changes in 'speed' can be seen by observing the lamp brightness.

Setup and adjustment

With the lamp connected, perform the following tests and adjustments. All voltage measurements below are with respect to the 0V test point.

1. Adjust VR3 for -7V at V_T.
2. Adjust VR1 for 0.5V at V_C.
3. Adjust VR7 so that LED3 is just lit, then back off anti-clockwise until the LED is off. V_S should measure 0.4-1.0V.
4. Adjust VR5 for a reading at V_{OVL} as close to 0V as you can manage.
5. Adjust VR6 for 4.1V at 'I_T'. This sets the motor overload threshold to 1.6 times its rating.

Now switch off the power, unplug the unit and restore the Live connection between terminals 4 and 5 of CON1.



The assembled module, ready for mounting in the case.

Also reconnect the crimp spade lead from CON5 to CON7. Make sure VR1 is set fully anti-clockwise and connect the lamp. Apply power and wait for RLY1 to switch off (indicated by LED4 switching off).

Check that the lamp begins to glow at low speed settings and reaches full brightness with the potentiometer fully clockwise. Once the operation is successful with the lamp, switch off the power and test it with the motor.

Verify that the motor speed can be controlled, noting that the motor will not start unless the speed potentiometer is rotated fully anti-clockwise first (LED4 off). Wait for RLY3 to be powered (LED5 lit) before bringing it up to speed.

Test the motor under load at around 25-50% of full speed and adjust the Torque trimpot, VR4, so that the motor does not drop markedly in speed when a load is applied.

Anti-clockwise rotation of VR4 increases the feedback control, meaning that more torque will be applied when the motor is under load. Too much speed compensation can cause the motor to speed up under load, so minor adjustments between tests are necessary to get it right.

Note also that the torque adjustment will affect the V_S value set with VR7, which ensures the PWM output is zero when the speed potentiometer is brought fully anti-clockwise. Check this by repeating steps 2 and 3 above after adjusting VR4.

Suppose the motor drops in speed too much under load even with maximum torque adjustment. In that case, the output from IC1b (current feedback) can be boosted by increasing the gain of this amplifier via clockwise adjustment of VR2. Again, make small adjustments between load tests.

Increasing the gain of IC1b will also require increasing the overload threshold (I_T) using VR6 by the same percentage.

Adding a reversing switch

If you need a motor reversing switch, you can use a 3PDT switch, as shown in Fig.8. One suitable switch is the "Lovato 3PDT 3 Position 60° Motor Reversing Cam Switch", rated at 20A. It has a knob actuator and is available from RS Components (Cat 8405624).

Two poles of the switch are used to reverse the motor polarity. The third switch pole ensures the motor is disconnected from power during the switching. It does this by opening the safety switch connection at terminals 7 and 8 of CON1. This prevents the motor from being switched into reverse while the motor is running.

After reversing, the motor can only be started once the speed pot is returned to its anti-clockwise position.

If a safety/emergency stop switch is also used, this will need to connect in series with the reversing switch pole at terminal 8 of CON1.

There is insufficient room inside the enclosure to install a reversing switch. Consequently, mains wiring for the motor connections and safety/emergency stop switch will need to run outside the enclosure using 10A sheathed mains cable, with the cables secured to the enclosure using cordgrip grommets.

The reversing switch must also be enclosed in an Earthed metal enclosure with cables secured using cordgrip grommets. Altronics H4280 grommets are suitable.

Note that if you have an on/off switch in series with the motor wiring, the switch needs to be a double-pole, double-throw type (DPDT) so that one pole connects or disconnects the power to the motor, with the second pole connected in the same way as shown for the third pole in the reversing switch. That way, the motor can't suddenly be reconnected, which could damage the Speed Controller. **PE**

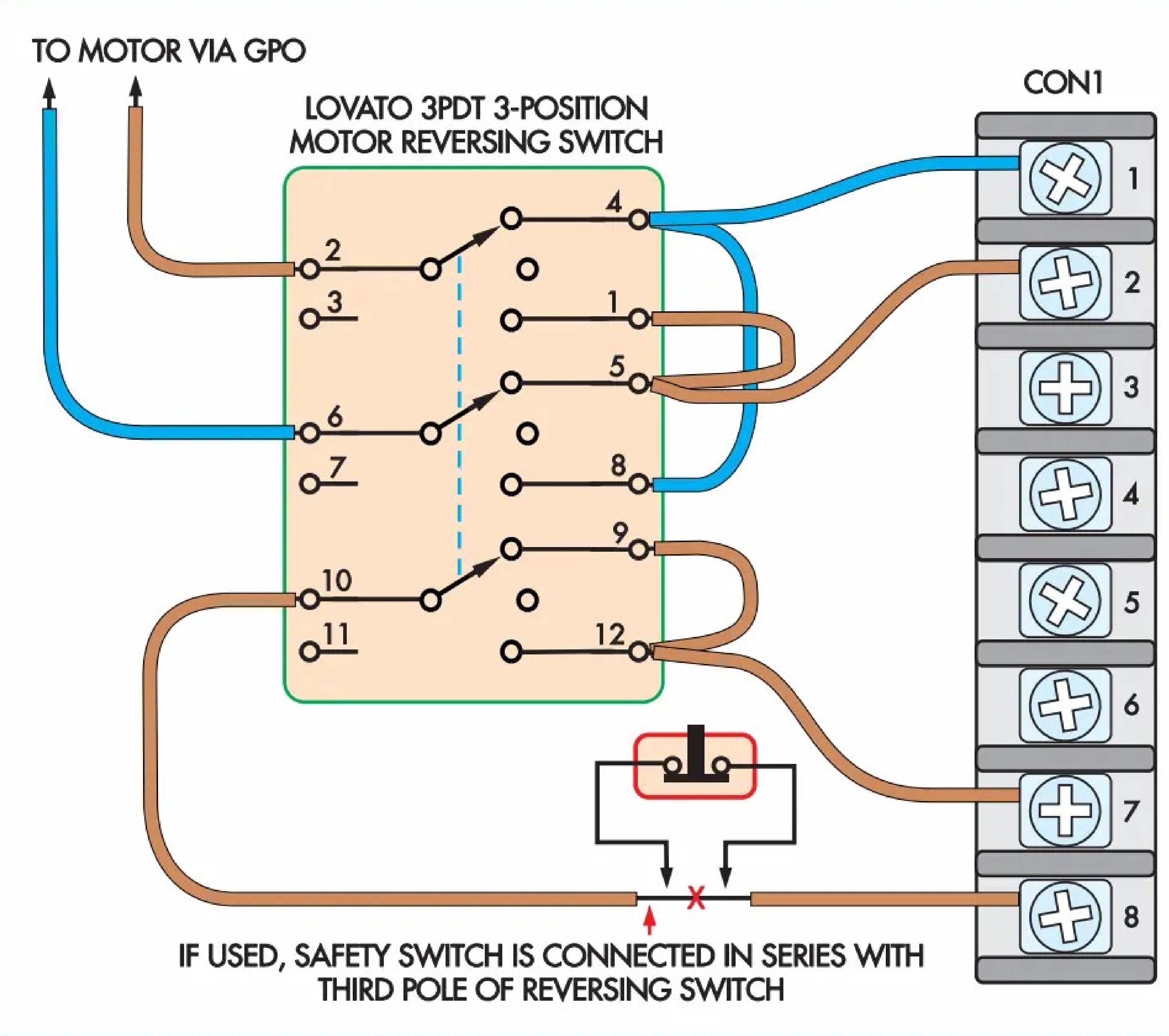


Fig.8: a 3PDT or 3P3T switch can be used to reverse the motor. The third pole (terminals 9-12) is used to shut down the Controller when the direction is changed. The speed pot needs to be reset to zero each time the switch is thrown before the motor will be powered again.

AOShop

The home for specialist audio, analogue and historic components – provided by Jake Rothman, *PE's Audio Out* columnist. The AOShop is your best bet for classic analogue 'NOS' (new old stock) components, including all parts for *Audio Out* projects and designs.

Ge semiconductors

Small-signal PNP transistors

NKT214F, OC57, OC59, 2N1377, 2N525	£1.00

Low-noise PNP transistors

GET106	£2.50

Small power PNP transistors

AC153, AC153K, AC188	£1.50

Small power NPN transistors

AC176, AC141K	£2.00

High-voltage PNP transistors

OC77, CV7001	£1.50

TO3 PNP power transistors

OC22, CV7054 (OC23), OC25, OC35, OC36, AD143, AD149, AD161, AD162	£2.00
AD140	£3.50
AD149, AD161/2 matched pair	£5.00

RF PNP transistors

OC41, OC42, 2SA12, 2SA53,AF124, AF178, GET872A	£1.50

NPN transistors

OC139, OC140, ASY73	£2.00

Diodes

CV7049 (OA10), CG92 (OA91)	£0.50

Si semiconductors

Diodes

ZC5800 RF Schottky	£0.20

Low-noise silicon transistors

BFW16A, 2SC3071, 2SC3068, 2SA1016K, 2SC2362K, 2SA970BL	£1.50
2SC2204, 2SD655, BC550C	£0.50
ZTX651	£0.30
ZTX751	£0.50

RF transistors (suitable for Theremin)

BF199	£0.50

Audio power MOSFETs

Exicon 10N20, 10P20	£6.50
Hitachi 2SJ99, 2SK343	£3.50
Hitachi 2SJ56, 2SK176	£8.50

JFETs

BFW11, BFW10, TIS73L, J177, J113, U1994, U1898, 2SJ176, J201	£1.00
J175, J176, J112, J111, 2N3820, 2N5467, BF244, 2N5460, J230	£0.60

Small power output/driver transistors

2SB649A, D669A, 2SA1208, 2SC2910, MJE253G, MJE243G, 2SA1725, 2SC4511	£1.20
BD139,BD140, BD135, BD156, BD435, BD436, MPSA63, BCV46	£0.50

MOSFETs

ZVP2106A	£0.30
Dual-gate 3SK45, BFS28	£1.50

Metal-cased transistors

BC143	£0.35
2N1711	£0.50
BCY71	£0.30
BC109C	£0.60

Amplifiers

LM318 high-speed op amp	£0.35
µA709 metal-cased op amp	£2.00
5534H metal-cased op amp	£3.00
LM384, TDA2030A, TDA2050V	£1.50
TAA435 (Mullard power amp driver for AD161/AD162)	£1.20

Synthesiser ICs

That/dbx 2180 VCA/VCF	£6.00
CA3080 VCA/VCF (vocoder)	£2.80
CA3280 VCA/VCF	£4.00
LM13600/ LM13700 VCA/VCF	£2.00
CA3086 transistor array	£1.00
PT2399 echo/delay	£1.50

Dual transistors

2N2639, 2N2223, 2N2910 (NPN)	£4.00
E401 (JFET Moog)	£4.00
2SK2145-Y dual JFET	£0.80
2N5564 JFET	£8.00
HN3C51F, HN3A51F	£1.00
DMMT3904/6, HN1A01F, HN1C01F	£0.50

Loudspeakers

PE Mini-Monitor

Volt PE165 6.5-inch woofer (each)	£85
Morel MDT29 tweeter (each)	£25
Kit *pair* of PE165/MDT29 plus Volt crossover parts and PCBs	£299
Monacor DT-28N tweeter (each)	£35
Vifa 19mm BC20SC15-04 tweeter (each)	£15

Volt crossover inductors

1.2mH, 1.5mH, 2mH, 2.7mH, 0.5mH (tapped at 0.3mH) (each)	£5.85

Fully assembled and tested high-quality speaker prototypes – ask for details

LS3/5As and other similar speaker systems (pair)	£200-£350

Fully tested reclaimed speakers

Vifa BC14 5-inch woofer (each)	£10
Vifa TC26 1-inch tweeter (each)	£10

Low-price speakers

Philips 4-inch 4070	£2.00
EMI 10x6-inch, 30Ω Alnico	£7
64mm 64Ω neodymium	£1.20

Capacitors

Note '10/63' denotes '10µF 63V'.

Polyester

3.3/100, 4.7/250, 4.7/63	£1.00
5.6/63, 8.2/63, 10/63	£2.00
Mullard 'Mustard' C296 0.22/400	£2.00

Polycarbonate

Axial 2.2/63 1%, 4.7/160, 6.8/63	£1.00

Radial

6.8/160V, 10/63	£2.00
Reclaimed 22/63	£2.00

Polystyrene

Philips 1% 4.7nF/160, 6.2nF/500, 12nF/63, 22nF/63, 110nF/63, 24nF, 2nF	£1.00
RIFA 1% 100nF/100, Suflex 90.9nF 0.5%	£2.00
Suflex 2.5% 10nF/63 (rad. or ax.)	£0.50

Radiation resistant

Siemens cellulose acetate MKL 2.2/25	£0.80

Electrolytic – Mullard blue 017 series

10/25, 22/25, 100/10	£0.50
150/40, 470/40, 1000/40	£1.00

Tantalum – axial metal cased

22/50, 47/35, 68/25, 100/20, 120/10, 150/16, 220/10, 330/6	£2.00
22/35, 33/35, 47/20, 68/15, 100/10, 150/6	£1.25
4.7/50, 6.8/35, 10/25, 10/35, 22/15	£1.00

Axial moulded-case tantalum

Kemet axial 6.8/10	£0.30
Kemet radial 33/10	£0.40
STC radial 100/20	£1.50

Tantalum bead

22/50	£1.00
470/3	£2.00
680/6.3	£3.00

Wet tantalum

220/25 axial	£3.20
Castanet button 140/30, 470/3	£3.20
Hughes 540/10	£3.20

Bipolar

Hermetic bipolar tantalum 16/35	£3.20
Elcap axial 10/50	£0.50
Generic radial 100/16, 470/35, 100/ 63, 22/35, 4.7/35, 220/16	£0.50

Philips solid-aluminium (axial) 121/123

47/16	£1.00
330/6.3	£2.00
100/35	£4.00

Philips Pearl 122 series (radial)

10/16	£0.30

Silvered mica (radial)

1nF/500 1%	£1.00

Trimmer capacitors

Vishay plastic-film 4-40pF	£0.80
Vishay plastic-film 5.5-45pF	£1.00
Vishay plastic-film 5-80pF	£1.20
Vishay plastic-film 10-250pF	£2.00
Mica 1-12pF, 2-40pF	£1.00

Audio transformers and inductors

Eagle transformers

LT44, LT722 driver, LT700, LT723 500Ω output	£2.50
LT30 500mW output	£3.50
5:1 interstage	£1.50

Repanco

T/T3 splitter transformer	£4.00
CH2 5mH RFC	£2.00

Balanced output transformer

Vigortronix 600Ω VTX-101-007	£10
Vigortronix 600Ω VTX-101-3001	£10
Vigortronix 600Ω VTX-101-3002	£15
Gardners 150Ω	£10
Reclaimed BBC LL74/MPC nickel core 600Ω	£12
Low-level 1:1 600Ω	£4
Reclaimed mic input transformer	£15

Inductors

82µH, 4.7mH, 100µH, 270µH, 10µH, 14µH (low Z)	£0.50
7-inch ferrite rod with MW and LW windings	£3.00

Special resistors

Bourns wire-wound trimmer 5kΩ 3059 JM panel-mount	£2.00
Thermistor RA53, R13	£4.00
A13	£2.00
Thermistor CZ1, CZ6	£1.50
Holco H2 2.2MΩ 1W, 1%	£1.00
Welwyn 1GΩ 2W	£1.00
5k Bourns 3321H cermet trimmer	£0.50

Potentiometers

Bourns 81

25kΩ lin cermet	£2.00
25kΩ lin, 5kΩ lin conductive plastic	£3.00

Bourns 91

10k dual-gang lin or log	£10.00

Plessey moulded-track

5kΩ log with switch	£3.00
50kΩ A/log	£3.00
Mil 250kΩ lin dual	£5.00

Alpha

16mm 4.7kΩ A/log	£0.80
220kΩ A/log	£0.80
10kΩ lin centre-detent dual-gang	£1.50

Allen Bradley J series/Honeywell

10kΩ lin 1 million cycles	£5.00
Blore Edwards AB 45 dual 5kΩ A/log with switch	£3.50
Alps RK9 dual-gang 5kΩ RD law	£4.00
Alps 50k log tapped motorised stereo	£5.00
BI P260 500kΩ log conductive plastic 1 million cycles	£2.00
Colvern wire-wound 100kΩ or 50kΩ dual-gang 3W	£5.00
Mil Spec hermetic 10Ω	£8.00

Miscellaneous

PCBs

Pocket Theremin (*EPE*, 1996)	£2.00
Elysian Theremin (*EPE*, 1996)	£6
Synth VCF, VCO (*EPE*, 2017)	£3.00
48V PSU (*EPE*, 2019)	£3.00
SMT dual transistor adapter	£0.40

Relays

STC hermetic 4190 24V DPDT	£4.00

Switches Blore Edwards/Plessey

MH1 4-pole 3-way	£3.00
Elma 01 2-pole 6-way	£6.00
3PDT guitar pedal stomp switch	£2.50

Valves

EC88, PCL84	£3.00

Contact

Jake Rothman
Manchester House
Market Street, Craven Arms
Shropshire SY7 9NN
(visit by appointment)

+44 (0)1597 829102

jrothman1962@gmail.com

Minimum order £5.00 inc post
Quantity discounts negotiable

Payment

PayPal, bank transfer, cheques (payable to "J Rothman", UK pounds only)
No VAT payable

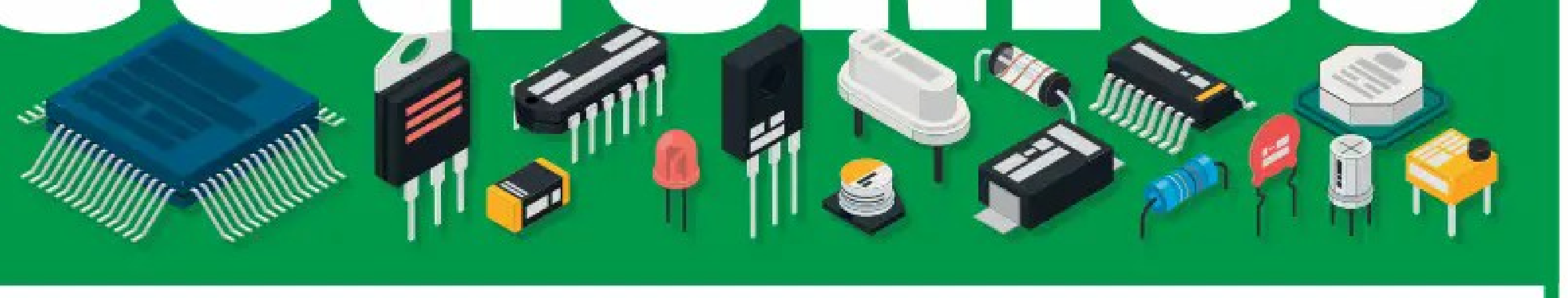
Postage

Small Jiffy bag £2.99
Small package £4.99
Big boxes and overseas at cost – ask for a quote

By Andrew Levido

Precision Electronics

Part 8: Voltage References



Last month, we looked at sampling and aliasing in DACs (digital-to-analog converters) and ADCs (analog-to-digital converters). Now we will describe how voltage references work, as they are critical to the precision of both ADCs and DACs.

We have covered a fair bit of ground so far in this Precision Electronics series, but we have not looked at the important topic of voltage references in detail. We mentioned them in our discussion of analog-to-digital and digital-to-analog conversion, but now is the time for a deeper dive.

In the very first article in this series, we discussed the difference between precision and accuracy. We learned that precision describes the repeatability or reliability of a measured quantity and is all about understanding and quantifying sources of error. On the other hand, accuracy is how closely a measured value matches the 'true' value (or an accepted proxy) of the quantity.

Most of what we have covered so far has been concerned with precision rather than accuracy. Voltage references are one of the places where these two come together. Put simply, voltage references provide a fixed voltage source at some defined value. This sounds simple, but there is a lot to unpack in that statement.

Initial accuracy – or not

The "defined value" part is where accuracy comes into play. A voltage reference is expected to provide a known output with some specific level of accuracy; that is, some measure of how close its output voltage is to the 'true' value.

This requirement implies the source has been calibrated somehow against a known standard. In many cases, this calibration is done for you at the factory, but sometimes you are expected to do it yourself.

An example of the former is the MAX6225ACSA reference we used in the DAC circuit in a previous instalment of this series. This is a 2.5V reference with an initial accuracy of ±200ppm (±0.02%). This means the voltage will be between 2.4995V and 2.5005V at 25°C – a figure achieved by calibration at manufacture.

Calibration would have been performed by trimming the chip's output voltage against a voltage standard with even higher accuracy. This standard will have been itself calibrated against an even better standard, and so on, in a chain ultimately traceable to the international standard definition of the volt.

Not all voltage references are trimmed to a standard voltage at manufacture. Some very expensive ultra-precision voltage references have woeful initial accuracy. The legendary LTZ1000, which will cost you the best part of $150, has an initial output voltage anywhere between 6.9V and 7.5V, but has extraordinary precision.

The intention is that you will calibrate the device in which you use this reference against some external standard, then take advantage of its incredible stability to ensure it stays that way.

Voltage reference errors

Whether you use a pre-calibrated or post-calibrated voltage reference, knowing its stability is critical. As you would expect, changes in temperature influence voltage references; achieving a stable voltage over temperature has been one of the driving forces behind the development of voltage reference technologies.

You will therefore see a figure for temperature drift in the voltage reference data sheet in the familiar absolute (V/°C) or relative (ppm/°C) units.

In many cases, you will also see a 'thermal hysteresis' figure. This is the maximum voltage change seen after cycling the device over some fixed temperature range – typically (but not always) from 25°C to 50°C and back. This is important because of the focus on accuracy; if the voltage changes slightly with temperature, we at least want it to return to a consistent value each time it reaches a given temperature.

To add to the growing list of errors, voltage references also drift with time, even if the temperature is held constant. Most precision voltage reference data sheets therefore include a figure for 'long-term stability' or similar. This can be expressed as an absolute or relative change in voltage per root thousand hours (ppm/√kh) or per thousand hours (ppm/kh).

The odd unit of root-kilohours is used because long-term stability typically has a logarithmic decay characteristic, with more drift in the first 1000 hours than in the second and so on.

Because of this early drift, some manufacturers of precision equipment using will 'burn in' the voltage references for a period before using them. This improves the long-term accuracy of the references (since the higher initial drift is done with), and allows the manufacturer to weed out any that show too much variation.

To put some context around these errors, consider the MAX6225ACSA. The initial accuracy was ±200ppm with a tempco (temperature coefficient) of ±2ppm/°C and a thermal hysteresis of 20ppm. The worst-case long-term drift is ±20ppm/kh. These are all pretty good figures, but the LTZ1000 is in a different class.

This latter chip includes an on-chip heater to keep the reference at a stable temperature, resulting in temperature drift in the ±0.05ppm/°C range. No thermal hysteresis figure is given, since the reference is always held at a constant temperature. Long-term stability is quoted as 0.28ppm/√kh.

I should note that building a device using the LTZ1000 or similar ultra-precision references is no trivial task. To get the most out of the chip, you must employ some ridiculously expensive high-precision, low-tempco resistor dividers and worry about all sorts

Part 8 – voltage references

of crazy details. Take this quote from the data sheet as an example:

The Kovar input leads of the TO-5 package form thermocouples when connected to copper PC boards. These thermocouples generate outputs of 35μV/°C. It is mandatory to keep the ... leads at the same temperature, otherwise 1ppm to 5ppm shifts in the output voltage can easily be expected from these thermocouples. Air currents blowing across the leads can also cause small temperature variations...

There is a whole online community dedicated to getting the best possible performance out of these and similar devices. Just search for "voltnuts".

Series and shunt references

Voltage references fall into two major categories: three-terminal (series) or two-terminal (shunt) devices, both of which are shown in Fig.1. Series references 'regulate' an input voltage to produce the reference output.

As you might imagine, the output is influenced by changes in the input voltage and by the load current, so for series regulators you will see figures in the data sheet for line and load regulation (ie, how stable the output is despite variations in input voltage and output current, respectively).

In the case of the MAX6225ACSA, the line regulation is ±7ppm/V for a V_{IN} above 10V and the load regulation is ±6ppm/mA. You should therefore ensure the input to a series voltage reference is well-regulated and keep the load current low and stable.

Achieving the latter can be a bit of a juggling act: you can add an op amp buffer to minimise the load current, but this will itself introduce errors that might outweigh those produced by the reference's load regulation. You have to crunch the numbers to work out what is optimal.

On the other hand, shunt references maintain a constant voltage drop while current is flowing through them. The voltage drop is influenced by the device current, and you can see from Fig.1 that this current is determined by both the source and the load currents.

Shunt references therefore usually include a single figure for output voltage change with current that encompasses both line and load regulation.

You should therefore aim to keep the device current constant if you want to achieve the best results with shunt-type references, most likely by employing an active current source and by keeping the load current fixed. Changes in shunt current with temperature will add to the shunt element's inherent temperature drift, so your current source needs to be relatively stable with temperature.

Zener references

The zener diode, with its well-defined reverse breakdown voltage, is the simplest form of shunt reference. This breakdown characteristic occurs because of two different mechanisms: the zener effect at low voltages, and the avalanche effect at higher voltages. The zener effect has a negative tempco, while the avalanche effect has a positive tempco.

As both these effects are present in zener diodes with breakdown voltages around 5-7V, it is possible to have these temperature effects more-or-less cancel each other out by careful selection of the diode and its operating conditions.

For example, a BZX55C5V1 5.1V, 400mW zener diode has a tempco of between +0.02% and -0.02% (±1mV/°C). By contrast, a BZX55C12 (12V) zener has a tempco of +0.11% (+13mV/°C). The initial accuracy of the 5.1V zener won't be great, but fed with a constant current, its voltage stability will be surprisingly good.

In fact, the LTZ1000 and MAX-6225ACSA both use internal zener diodes as the basis of their reference.

You can temperature-compensate a zener diode by putting it in series with a forward-biased standard diode, as long as you choose a zener with a tempco of about +2mV/°C. That's because the tempco of a regular diode's forward drop is about -2.1mV/°C.

Otherwise, you can buy temperature-compensated zener references like the LM329, or even the LM399, which includes an on-chip heater.

We mentioned above that you need to maintain a constant current through a zener reference. For example, the BZX55C5V1 has a dynamic resistance of up to 35Ω, so a change in bias current of just 1mA will shift the output voltage by 35mV – as much as a temperature rise of 35°C.

Fig.2 shows a clever circuit that uses the zener itself to provide a stable bias current and allows the output voltage to be adjusted to boot. The zener voltage is amplified by the non-inverting amplifier to produce an output voltage, $V_{OUT} = V_Z (1 + R1 \div R2)$. The output voltage is then used to establish the zener bias current via R3, $I_Z = (V_{OUT} - V_Z) \div R3$.

You must use a single supply with this circuit – with a split supply, it could settle into a second stable state with the zener forward-biased. If you are worried about start-up, you can add a 1MΩ or greater resistor from the zener's cathode to the positive supply.

You will often see the term 'buried zener' to describe precision zener references. This just means that the diode junction (where the reverse breakdown occurs) is formed below the surface of the semiconductor and covered with a layer of diffusion material, resulting in a more stable device with lower noise.

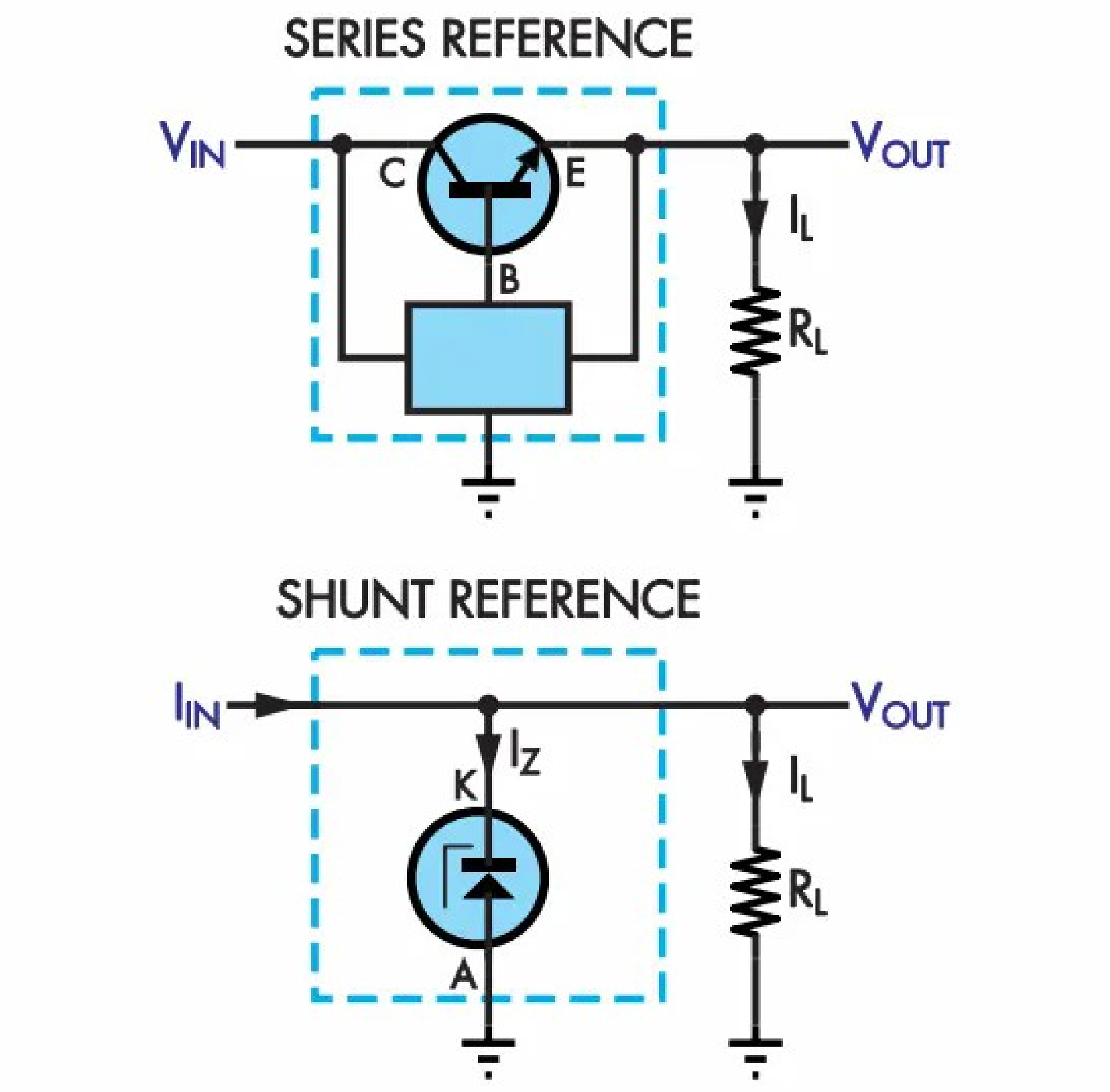


Fig.1: voltage references are available as either three-terminal series pass devices or two-terminal shunt devices. In either case, keeping the input voltage or current (and the output current) constant is critical to getting the best accuracy.

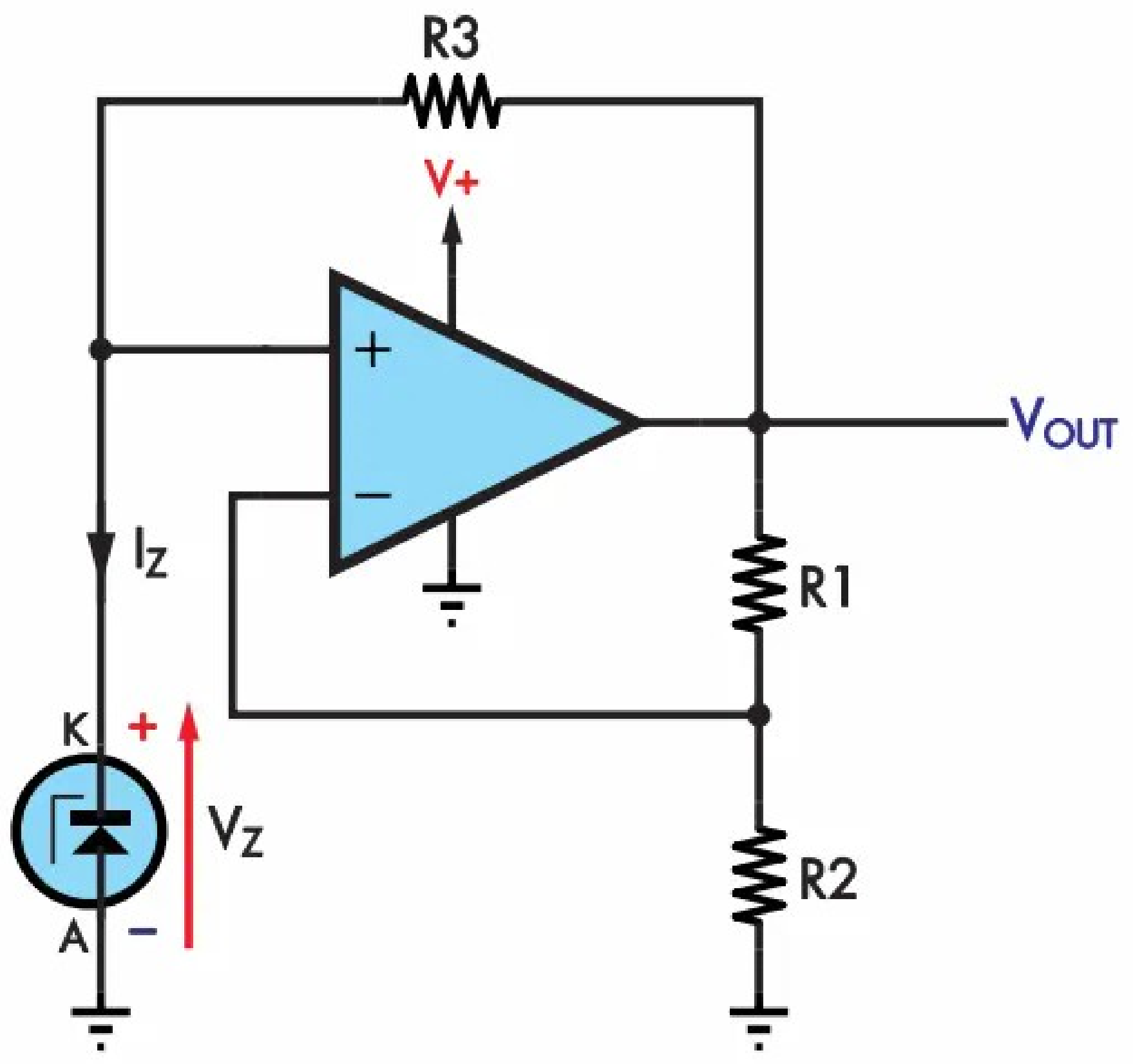


Fig.2: in this circuit, the zener bias current is derived from its own stable output. R1 and R2 allow the output voltage to be amplified if necessary.

Band-gap references

Band-gap voltage references are

much more common than zener references these days, especially inside integrated circuits. Pretty much every voltage regulator, linear or switching, uses one.

Developing a semiconductor voltage reference with a low tempco was no trivial task, but it came down to adding a voltage with a positive tempco to another voltage with a negative tempco just like the compensated zener.

The idea was first used commercially by Bob Widlar in 1971. Bob Widlar was an erratic genius who pioneered the first commercially successful op amps, comparators and three terminal-voltage regulators, among many others.

Part of his success was to recognise and work with the strengths of the IC production process. He understood that it was very hard to manufacture components of precise absolute value in silicon, but was relatively easy to make components with precisely matched values.

The Ebers-Moll large-signal BJT model (a really useful model I strongly recommend you study) tells us that the base-emitter voltage (V_{BE}) of a transistor is related to its collector current (I_C) by the relationship $V_{BE} = V_T \log_e(I_C \div I_S)$. V_T is the thermal voltage (proportional to absolute temperature) and I_S is the reverse saturation current (also highly temperature-dependant).

With a fixed collector current, the base-emitter voltage has a negative tempco of -2.01mV/°C, since the tempco of the I_S term dominates. However, the difference between the base-emitter voltages of two transistors with different collector current densities has a positive tempco.

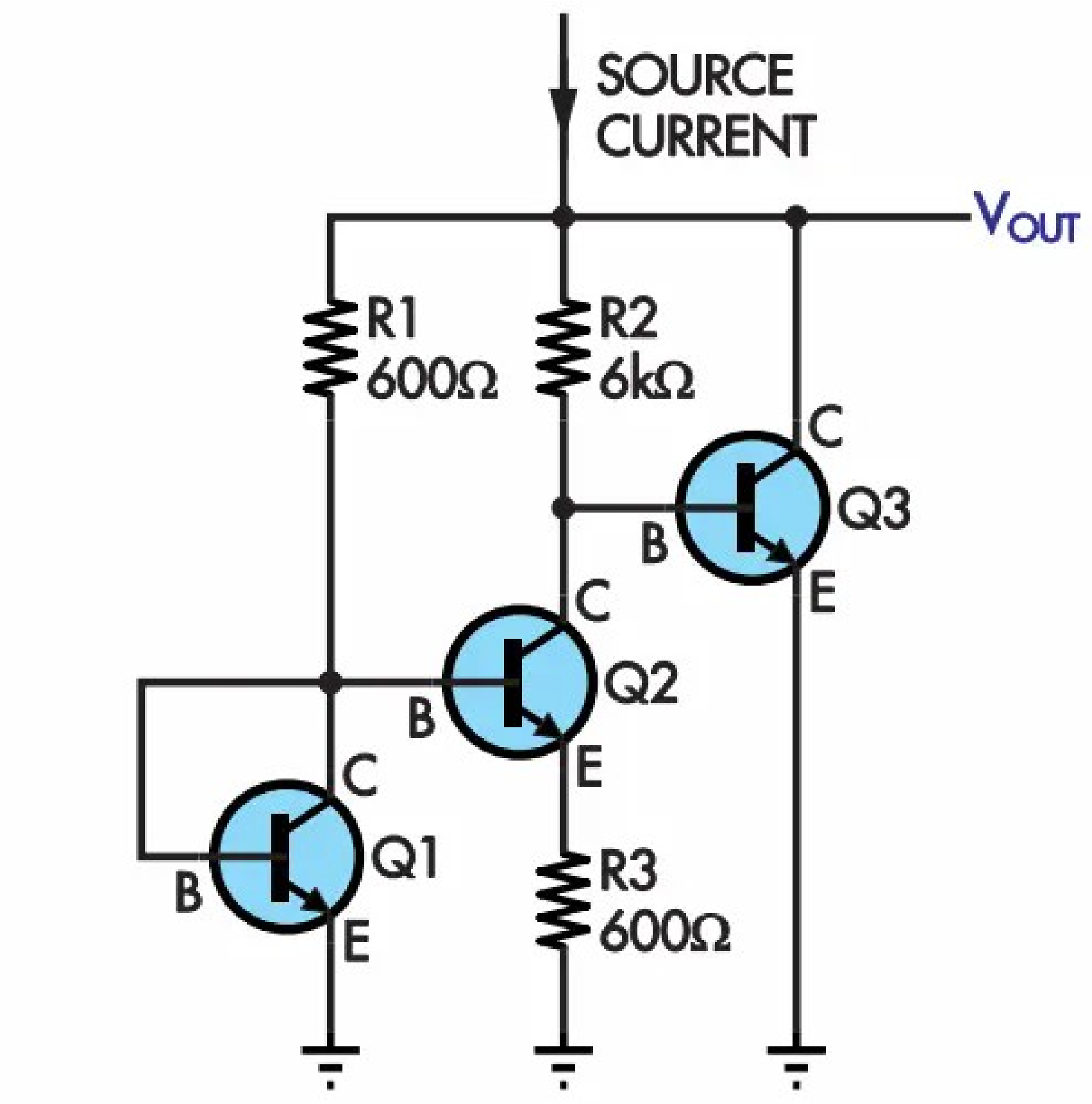


Fig.3: the simplest band gap reference consists of just three transistors. The voltage across R3 ($V_{BE1} - V_{BE2}$), and therefore that across R2, has a positive tempco, offsetting the negative tempco of Q3's base-emitter voltage.

This can be achieved by using two identical transistors with different collector currents. If you are mathematically inclined, it is pretty easy to see why by simplifying the expression $V_{BE1} - V_{BE2} = V_T(\log_e[I_{C1} \div I_S] - \log_e[I_{C2} \div I_S])$. This becomes $V_{BE1} - V_{BE2} = V_T \times \log_e(I_{C1} \div I_{C2})$. Note that I_S has disappeared, leaving V_T with its positive tempco the only temperature-dependent term.

Fig.3 shows how this phenomenon can be used in practice. Identical transistors Q1 and Q2 have collector currents in a 10:1 ratio because of the values of R1 and R2. Both transistors have the same base voltage, so the voltage at the emitter of Q2 must be $V_{BE1} - V_{BE2}$. The 10:1 I_C ratio means this voltage will be $2.3V_T$ (around 60mV).

The voltage across R2 will therefore be $23V_T$ or about 600mV. The output voltage, V_{OUT}, will be this voltage plus the V_{BE} drop in transistor Q3 for an output of around 1.2V.

Importantly, the tempco of V_{OUT} will be that of V_{BE3}, around -2.01mV/°C, plus 23 times that of V_T (23 × +86.2μV/°C = +1.98mV/°C), resulting in an overall tempco of -27.4μV/°C, which is around 10ppm per °C. Not bad for three transistors.

This type of reference is called a bandgap reference because its output voltage for zero tempco corresponds with the theoretical bandgap voltage of the semiconductor material (~1.14V for silicon).

Paul Brokaw developed an improved circuit in 1974, overcoming some of the limitations of the Widlar circuit, which could only produce a 1.2V output and required a fairly constant supply current. Brokaw's circuit is shown in Fig.4.

Brokaw's circuit uses transistors with identical collector currents (due to identical R3s), but with differing physical on-chip areas to achieve different current densities. The voltage across R1 is the difference in base-emitter voltages, $V_T \times \log_e(N)$ where N is the ratio of transistor areas. The voltage across R2 is therefore $V_T \times \log_e(N) \times 2(R2 \div R1)$, which has a positive tempco due to V_T.

The reference output voltage will be this voltage plus the base-emitter voltage of Q1 with its negative tempco. By choosing the right values for R1, R2 and N, you can cancel the temperature dependencies as we did before. Fig.5 shows the same circuit configured to provide output voltages higher than the nominal 1.2V bandgap voltage.

You can buy off-the-shelf bandgap references for around 60¢ each. The Microchip LM404x series, for example, are packaged as shunt references and available with output voltages from 1.225V to 5.000V. The 2.5V version in C grade has an absolute accuracy of ±0.5%.

The versatile and very popular TL431 is even cheaper. This is also a 2.5V reference with a ±0.5% initial accuracy (C grade) with a typical tempco of ±10ppm/°C. You frequently see these devices in the voltage regulation circuits of low-cost flyback switch-mode power supplies.

Many of our projects have used them too. For example, the DC Supply Protectors project from the May 2025 issue used one to set the over-voltage threshold. The 500W Power Amplifier published in the April, May & June 2023 issues of *PE* also used two, as part of the load-line protection circuitry.

Exotic references

Another more exotic voltage reference technology is the 'JFET pinch-off' reference. These work on a similar principle to the bandgap reference in that the difference in pinch-off voltage of two JFETs has a negative tempco that offsets the positive tempco of a current source.

The ADR420 reference uses this technique to achieve an initial accuracy of ±400ppm (B grade) with a tempco of ±3ppm/°C. Its long-term stability is 50ppm/kh. The big advantage of this type of reference over bandgap references is their very low noise.

Another interesting reference technology is the floating gate reference. These rely on a Mosfet (actually an array of Mosfets) with a well-insulated 'floating' gate. At manufacture, charge is applied to the gate which, like a capacitor, charges to a particular voltage.

The Mosfet then acts as a high-impedance voltage follower to read out this voltage. The voltage will remain stable as long as the gate charge does not change.

The only commercial examples I am aware of are in the ISL2090 series. The 2.50V B-grade version has an initial accuracy of ±0.02%, a tempco of ±7ppm/°C and long-term stability of 20ppm/kh. By my calculations (assuming a gate capacitance of 100pF charged to 1V), the gate leakage must be something less than 12 electrons per hour! Amazing.

Voltage reference noise

Depending on your application, you may have to take voltage reference noise into account. Unlike op amps, there is little consistency in how manufacturers specify the noise in their voltage references, especially at very low frequencies (say, below 10Hz).

This is the area we usually operate in, and it is the territory where $^{1}/_{f}$ noise tends to dominate, meaning we can't just extrapolate from a wideband noise voltage to a noise density figure.

Because we are dealing with DC signals, we can almost always add some filtering to reduce the voltage noise. Many references come with a 'noise reduction' pin that you bypass to ground with a small capacitor to improve the noise performance.

For example, the MAX6225ACSA has such a pin which, if bypassed with a 1µF capacitor, will reduce the noise density above 100Hz from around 40nV/√Hz to under 15nV/√Hz.

If we add an external filter, we have to make sure it does not adversely influence the reference voltage. Fig.6 shows one example of how we could do this. R1 and C1 form a low-pass filter with a -3dB frequency of 0.016Hz. The bottom end of C1 is bootstrapped by R2/C2 so that the voltage across C1 is zero in the steady state.

If we did not do this, the leakage current through the capacitors would cause a voltage error as it is dropped across R1. You should also use a low-noise precision op amp buffer so that you don't add new errors.

Miscellany

Precision voltage references can be expensive, so it is worth treating them with respect. Below are a few considerations you may need to be aware of, depending on your application.

Solder shift: the worst thermal shock a voltage reference is likely to experience is during the assembly process, particularly if it is reflowed onto the printed circuit board (PCB). Like thermal hysteresis, this can cause a permanent change in the output voltage, known as 'solder shift'.

You generally won't find information on this in the data sheets, but there are a few app notes out there that discuss it. It is only going to be of concern for very high precision applications but is worth knowing about. You might want to hand-solder the reference if your application falls into this category.

Start-up time: many references use their own output to provide stabilised bias conditions (like the circuit in Fig.2), so they include start-up circuitry to ensure everything comes up in an orderly fashion. This means that the output voltage might not reach its final stable value for some time.

It is not unusual for this time to be tens or even hundreds of milliseconds. Your application should be aware of this and not use the reference until it is stable.

Board flex: flexing a PCB that contains a precision reference can produce a measurable change in the output. One manufacturer suggests this can be as much as a 60ppm peak-to-peak change for a reference mounted on a standard 1.6mm FR4 board that is 100mm wide board and flexed up and down by 1.8mm.

You can minimise such mechanical stress by fixing boards down firmly and/or by using slots in the board to mechanically isolate the reference.

Leakage: leakage currents across the surface of a printed circuit board can cause errors in precision references. References with noise-filtering pins (eg, the MAX6225ACSA) can be especially vulnerable, since these usually expose a high-impedance summing node to the outside world.

A few tens of nano amps flowing into or out of one of these nodes can shift the output voltage by hundreds of ppm. Flux contamination or skin oils are more than enough to allow this level of current leakage, so it pays to clean your precision boards thoroughly and to keep your fingers off them once you have.

Conclusion

Precision voltage references are unique in that they are one component that combines both precision and accuracy, allowing the device they are used in to deal in absolute quantities.

In the next and final article of this series, we will zoom out and look at the big picture – how one might go about the high-level design of a precision electronics device from a whole-system perspective.

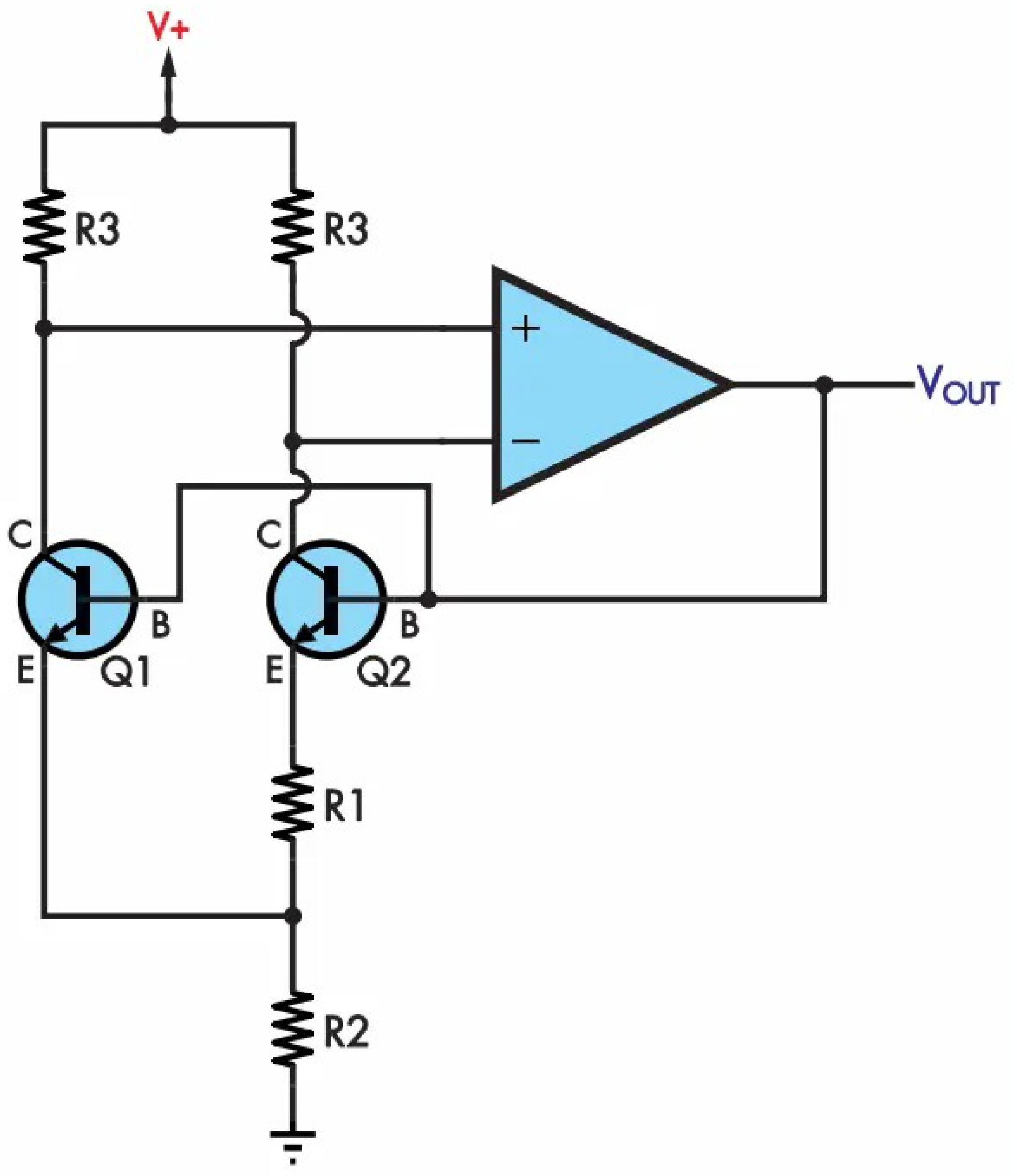


Fig.4: the Brokaw band gap reference uses two transistors with the same collector currents, but of differing areas to produce a voltage across R2 with a positive tempco to offset the negative tempco of Q1's base-emitter voltage.

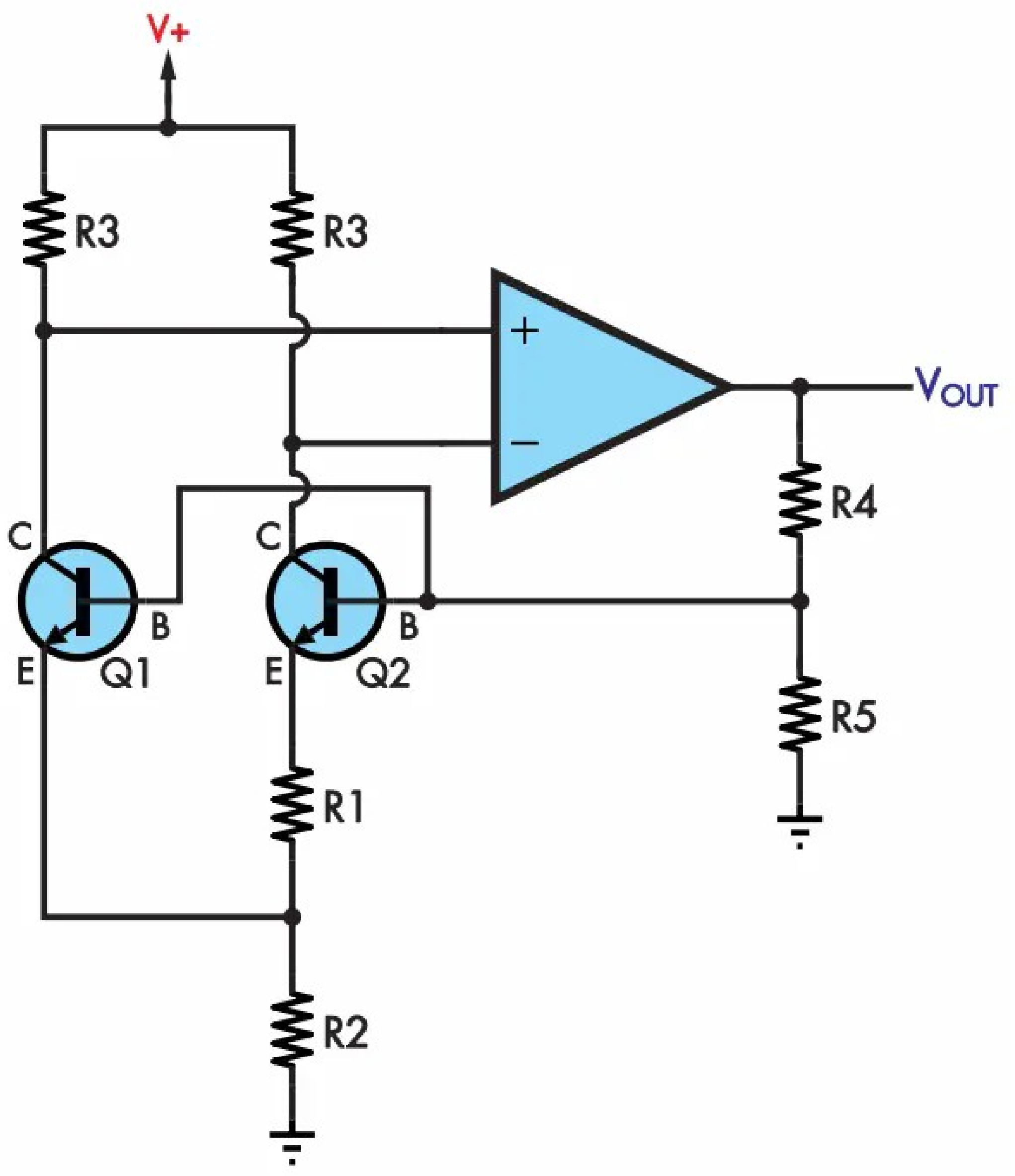


Fig.5: the Brokaw band gap reference can easily be adapted to produce higher output voltages.

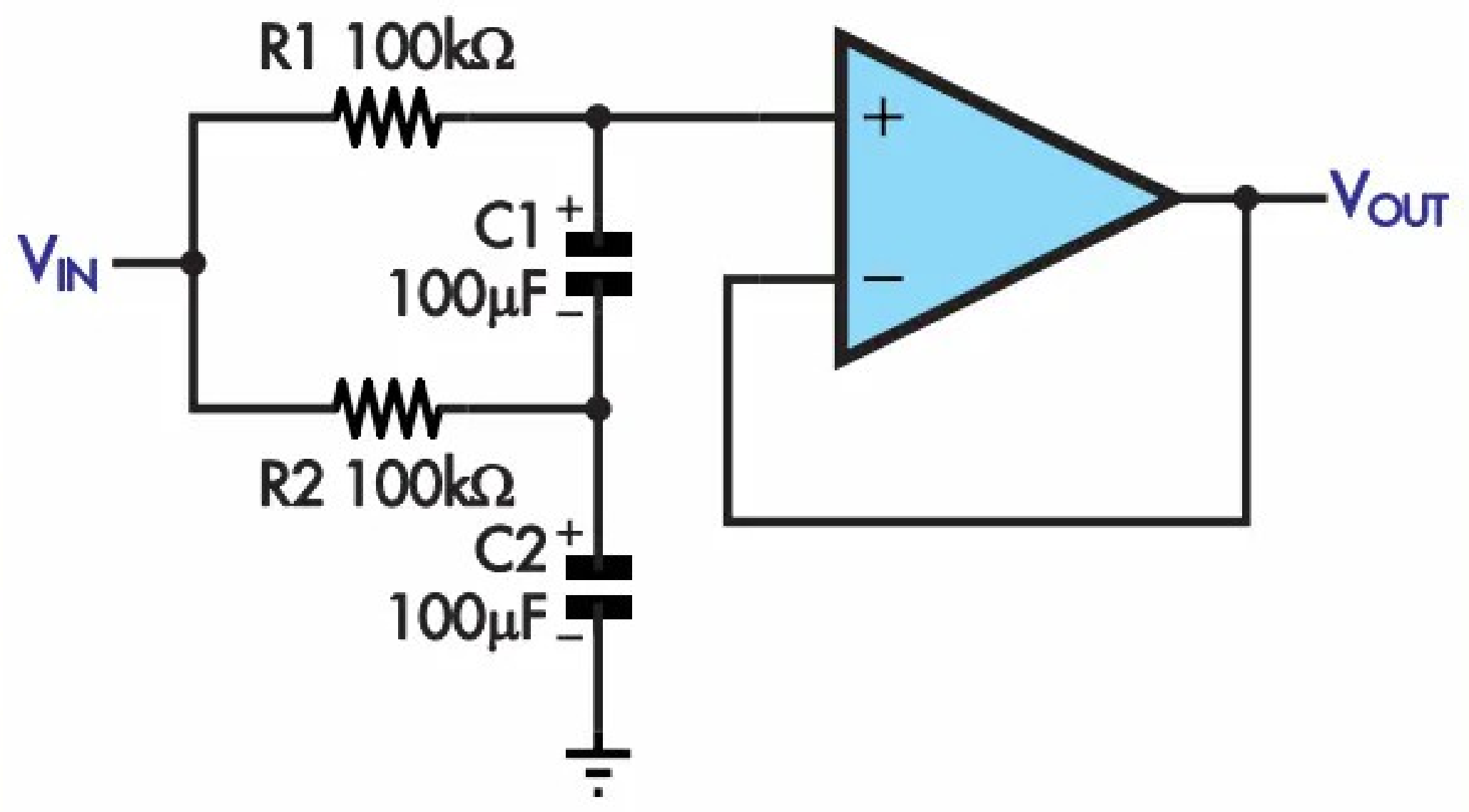


Fig.6: this RC filter reduces the noise voltage produced by a voltage reference by limiting the bandwidth to 0.016Hz. R1 and C1 are the filter, while R2 and C2 bootstrap the bottom of C1 to eliminate its leakage current, which would be otherwise be dropped across R1, causing the reference voltage to drop.

Practical Electronics PCB SERVICE

PROJECT	CODE	PRICE
AUGUST 2025		
Micromite Explore-40 PCB	07106241	£2.99
Micromite Explore-40 Audio Breakout PCB	07101222	£2.99
8-Channel Learning IR Remote Receiver PCB	15108241	£5.99
FlexiDice PCB set	08107241-2	£4.99
JULY 2025		
Compact OLED Clock & Timer PCB	19101231	£4.99
180-230V DC Motor Speed Controller PCB	11104241	£9.99
Mains Power-Up Sequencer PCB (same as Feb '25)	10108231	£11.99
JUNE 2025		
Styloclone (case version)	23106241	£6.99
Styloclone (standalone version)	23106242	£7.99
Dual Mini LED Dice (TH LEDs, white solder mask)	08103241W	£2.99
Dual Mini LED Dice (TH LEDs, black solder mask)	08103241B	£2.99
Dual Mini LED Dice (SMD LEDs, white solder mask)	08103242W	£2.99
Dual Mini LED Dice (SMD LEDs, black solder mask)	08103242B	£2.99
Dual Hybrid Supply regulator board	18107211	£5.99
Dual Hybrid Supply front panel control board	18107212	£2.99
Dual Hybrid Supply CPU board	01106193	£3.99
Dual Hybrid Supply LCD adaptor	01106196	£2.99
MAY 2025		
Skill Tester 9000	08101241	£9.99
USB-C Serial Adaptor (black solder mask)	24106241	£2.99
DC Supply Protector (SMD adjustable version)	08106241	£3.99
DC Supply Protector (TH adjustable version)	08106242	£3.99
DC Supply Protector (TH fixed version)	08106243	£3.99
APRIL 2025		
ESR Test Tweezers PCB set	04105241+	£5.99
Fan Speed Controller Mk2	10104241	£4.99
Calibrated Microphone (SMD version)	01108231	£3.99
Calibrated Microphone (TH version)	01108232	£3.99
MARCH 2025		
Pico Gamer	08104241	£9.99
Compact Frequency Divider	04112231	£4.99
FEBRUARY 2025		
Laser Communicator transmitter	16102241	£3.99
Laser Communicator receiver	16102242	£3.99
Model Railway Points Controller transmitter	09101241	£3.99
Model Railway Points Controller receiver	09101242	£2.99
Mains Power-Up Sequencer	10108231	£11.99
JANUARY 2025		
USB to PS/2 Converter for VGA PicoMite	07111231	£3.99
ps2x2pico USB to PS/2 Converter	07111232	£3.99
Compact Microphone Preamplifier (case version)	01110231	£5.99
Compact Microphone Preamplifier (integrated version)	01110232	£5.99
DECEMBER 2024		
Raspberry Pi Clock Radio PCB pair	19101241/2	£12.99
VGA PicoMite (blue)	07107221	£5.99
Secure Remote Switch receiver	10109231	£4.99
Secure Remote Switch transmitter (module version)	10109232	£2.99
Secure Remote Switch transmitter (discrete version)	10109233	£2.99

PROJECT	CODE	PRICE
NOVEMBER 2024		
Ideal Diode Bridge Rectifiers (28mm square)	18101241	£3.99
Ideal Diode Bridge Rectifiers (22mm square)	18101242	£3.99
Ideal Diode Bridge Rectifiers (4-pin SIL)	18101243	£3.99
Ideal Diode Bridge Rectifiers (tiny SOT-23)	18101244	£3.99
Ideal Diode Bridge Rectifiers (standalone SMD)	18101245	£4.99
Ideal Diode Bridge Rectifiers (standalone TH; 2oz copper)	18101246	£6.99
Multi-channel Volume Control (volume PCB)	01111221	£4.99
Multi-channel Volume Control (control PCB)	01111222	£4.99
Multi-channel Volume Control (OLED PCB, black)	01111223	£4.99
OCTOBER 2024		
TQFP-32 Programming Adaptor	24108231	£4.99
TQFP-44 Programming Adaptor	24108232	£4.99
TQFP-48 Programming Adaptor	24108233	£4.99
TQFP-64 Programming Adaptor	24108234	£4.99
2m VHF FM Test Signal Generator	06107231	£4.99
SEPTEMBER 2024		
PIC Programming Adaptor	24105231	£4.99
Salad Bowl Speaker crossover	01109231	£7.99
AUGUST 2024		
Arduino ESR Meter combined L/C-ESR	04106182	£9.99
Arduino ESR Meter separate ESR	04106183	£7.99
Watering System Controller	15110231	£11.99
JULY 2024		
Dynamic NFC Tag (22 × 31mm)	06101231	£5.95
Dynamic NFC Tag (86 × 54mm)	06101232	£7.95
Reciprocal Frequency Counter	CSE230101C	£8.95
Reciprocal Frequency Counter front panel (black mask)	CSE230102	£7.95
Pi Pico-based Thermal Camera	04105231	£6.95
JUNE 2024		
Loudspeaker Test Jig	04106231	£12.95
MAY 2024		
GPS-Disciplined Oscillator	04103231	£12.95
Dual RF Amplifier	CSE220602A	£5.95
Songbird	08103231	£7.95
APRIL 2024		
Basic RF Signal Generator	CSE221001	£8.99
Basic RF Signal Generator front panel (black solder mask)	CSE220902B	£9.99
Silicon Chirp Cricket	08101231	£7.99
Test Bench 'Swiss Army Knife'	04110221	£9.99
MARCH 2024		
Digital Volume Control Potentiometer SMD version	01101231	£8.95
Digital Volume Control Potentiometer through-hole version	01101232	£9.95
Advanced SMD Test Tweezers body	04106221	£6.95
Advanced SMD Test Tweezers legs (pair)	04106212	£7.95
FEBRUARY 2024		
Active Mains Soft Starter	10110221	£10.95
JANUARY 2024		
Q Meter	CSE220701	£8.95
Q Meter front panel (black solder mask)	CSE220704	£7.95
Raspberry Pi Pico W BackPack	07101221	£8.95
DECEMBER 2023		
Digital Boost Regulator	24110224	£8.95
Dual-Channel Power Supply for Breadboards	04112221	£11.95
Display Adaptor for the Breadboard PSU	04112222	£9.95

PCBs for most recent *PE/EPE* constructional projects are available. From the July 2013 issue onwards, PCBs with eight-digit codes have silk screen overlays and, where applicable, are double-sided, have plated-through holes, and solder mask. They are similar to photos in the project articles. Earlier PCBs are likely to be more basic and may not include silk screen overlay, be single-sided, lack plated-through holes and solder mask.

Always check price and availability in the latest issue or online. A large number of older boards are listed for ordering on our website.

In most cases we do not supply kits or components for our projects. For older projects it is important to check the availability of all components before purchasing PCBs.

Magazine back issues are available – see https://www.electronpublishing.com/product-category/pe-back-issues-print-and-digital/

Double-sided | plated-through holes | solder mask

PROJECT	CODE	PRICE
NOVEMBER 2023		
LC Meter Mk3	CSE220503C	£9.95
DC Supply Filter for vehicles	08108221	£8.95
Discrete Audio Op Amp PCB	AO1-JUL23	£9.95
Discrete Audio Op Amp PCB + essential components	AO2-JUL23	£17.95
OCTOBER 2023		
Buck/Boost Charger Adaptor	14108221	£9.95
PIC Breakout Board for SOIC parts	24110225	£5.95
PIC Breakout Board for DIP parts	24110222	£5.95
AVR64DD32 Breakout board	24110223	£5.95
Automatic Train Controller	09109221	£6.95
Chuff Sound module	09109222	£6.95
SEPTEMBER 2023		
Mini LED Driver	16106221	£7.95
New GPS-Synchronised Clock	19109221	£9.95
AUGUST 2023		
110dB RF Attenuator	CSE211003	£9.95
Universal Battery Charge Controller (2023 update)	14107192	£9.95
Wide-Range OhmMeter	04109221	£12.95
JULY 2023		
Multimeter Checker/Calibrator	04107221	£10.95
JUNE 2023		
Arduino Programmable Load	04105221	£8.95
Buck-Boost LED Driver	16103221	£8.95
MAY 2023		
Precision AM-FM DDS Signal Generator	CSE211002	£14.95
Improved SMD Test Tweezers programmed PIC	0410621PIC	£12.95
AO 2x-dual-to-quad through-hole (pack of five)	AO1-MAY23	£6.95
AO 2x-dual-to-quad SMD (pack of five)	AO2-MAY23	£6.95
AO 2x-single-to-dual through-hole (pack of five)	AO3-MAY23	£6.95
AO 2x-single-to-dual SMD (pack of five)	AO4-MAY23	£6.95
APRIL 2023		
500W Amplifier Module	see p.22, April 2023	
Clipping Indicator (per channel)	01112211	£7.95
CD Welder Power Supply (one needed)	29103221	£9.95
CD Welder Controller (one needed)	29103222	£9.95
CD Welder Energy Storage module (several needed)	29103223	£7.95
AO Universal Dual Op Amp Board	AO1-APR23	£9.95
AO Stereo RIAA precision passives kit for dual op amp	AO2-APR23	£8.95
MARCH 2023		
Pico BackPack	07101221	£9.95
Semaphore Signal (controller)	09103221	£7.95
Semaphore Signal (blade)	09103222	£5.95
FEBRUARY 2023		
Tesla Coil driver board	26102221	£9.95
Tesla Coil potentiometer board	26102222	£5.95
Cooling Fan Controller & Loudspeaker Protector	01102221	£8.95
Remote Gate Controller	11009121	£12.95
JANUARY 2023		
Classic LED Metronome – 8-LED	23111211	£7.95
Classic LED Metronome – 10-LED	23111212	£8.95
Multi-Channel Speaker Protector – 6 channel	01101221	£9.95
Multi-Channel Speaker Protector – 4 channel	01101222	£7.95
Remote Control Range Extender – IR-to-UHF	15109212	£5.95
Remote Control Range Extender – UHF-to-IR	15109211	£7.95
AO Universal Single Op Amp Board	AO1-JAN23	£7.95

PROJECT	CODE	PRICE
DECEMBER 2022		
Hummingbird Amplifier	01111211	£9.95
SMD Trainer PCB	29106211A	£8.95
SMD Trainer PCB + parts	29106211B	£13.95
NOVEMBER 2022		
USB Cable tester – main PCB	04108211	£12.95
USB Cable tester – front panel	04108212	£5.95
USB Cable tester – optional panel	SC5970	£5.95
Model Railway Carriage Lights – PCB	09109211	£6.95
AO trnsfmr PCB – standard VTX-A range	VTX-101-007	£6.95
AO trnsfmr PCB – dual-outline VTX102-3001/101-3002	VTX-Dual	£6.95
OCTOBER 2022		
SMD Test Tweezers – PCB and pair of tweezer arms	04106211-2	£11.95
SMD Test Tweezers – programmed PIC12F1572-I/SN	0410621A	£7.95
Tele-com	12110211	£12.95
SEPTEMBER 2022		
Touchscreen Digital Preamp – main board	01103191	£12.95
Touchscreen Digital Preamp – adaptor board pair	01103192	£5.95
20A DC Motor Speed Controller	11006211	£9.95

For the many earlier PCBs that we stock please see the PE website: www.electronpublishing.com

PE/EPE PCB SERVICE

Order Code	Project	Quantity	Price

Name ....

Address ....

....

Tel ....

Email ....

Card No ....

Valid From .... Expiry Date ....

Card Security No ....

You can also order PCBs by phone, email or via the shop on our website: www.electronpublishing.com

No need to cut your issue – a copy of this form is just as good!

All prices include VAT and UK p&p. Add £4 per *project* for post to Europe; £5 per *project* outside Europe.

Orders and payment should be sent to:

Practical Electronics, Electron Publishing Ltd
113 Lynwood Drive, Merley, Wimborne, Dorset BH21 1UU
Tel 07918 614662 Email: pe@pemag.au
Online Shop: www.electronpublishing.com

Sorry, cheques cannot be accepted as we are based in Australia and our UK bank account does not allow cheque deposits.

NOTE: Most boards are in stock and sent within seven days of order receipt. Please allow up to 28 days delivery if we need to restock.

Practical Electronics PCB SERVICE

PROJECT	CODE	PRICE
AUGUST 2022		
Multi-purpose Battery Manager – I/O Expander	11104212	£5.95
Multi-purpose Battery Manager – Switch Module	11104211	£8.95
Simple MIDI Music Keyboard (for 8 switches)	23101213	£6.95
Nano Pong	08105212	£7.95
JULY 2022		
Silicon Labs AM/FM/SW Radio	CSE210301C	£10.95
Level Crossing Controller	09108211	£6.95
JUNE 2022		
Full-wave Motor Speed Controller	1010221	£8.95
PIC Programming Helper for 8-pin PICs only	24106211	£7.95
PIC Programming Helper for 8, 14 or 20-pin PICs	24106212	£10.95
Advanced GPS Computer	05102211	£9.95
MAY 2022		
Bus board PCB for Analogue Vocoder	AO1-MAY22	£10.95
Complete set of 14 PCBs for Analogue Vocoder	AO2-MAY22	£97.95
Programmed EEPROM for Digital FX Unit	FX1-MAY22	£10.95
Programmed PIC for Digital FX Unit w/ potentiometer	FX2-MAY22	£8.95
APRIL 2022		
64-key MIDI Matrix shield	23101211	£8.95
64-key MIDI Matrix switch board	23101212	£11.95
High-current Battery Balancer	14102211	£10.95
Digital FX Unit – using potentiometer	01102211	£9.95
Digital FX Unit – using BCD switch	01102212	£9.95
Universal Audio PSU	AO1-APR22	£11.95
MARCH 2022		
Mini Isolated Serial Link	24102211	£5.95
Busy Loo Indicator	16112201	£5.95
Analogue Vocoder – Band-pass filter board	AO1-MAR22	£9.95
Analogue Vocoder – HP/LP filter board	AO2-MAR22	£9.95
FEBRUARY 2022		
Arduino-based Power Supply	18106201	£9.95
Battery Monitor Logger	11106201	£10.95
Electronic Wind Chimes	23011201	£10.95
Analogue Vocoder – Driver Amplifier	AO-FEB22	£8.95
JANUARY 2022		
Vintage battery Radio Li-ion Power Supply	11111201	£9.95
MiniHeart: A Miniature Heartbeat Simulator	01109201	£8.95
DECEMBER 2021		
AM/FM/SW Digital Receiver	CSE200902A	£13.95
Balanced Input and Attenuator for USB CODEC	01106202	£11.95
NOVEMBER 2021		
Dual Battery Lifesaver	11111202	£6.95
OCTOBER 2021		
Mini Wi-Fi LCD BackPack	24106201	£8.95
SEPTEMBER 2021		
USB SuperCodec PCB	01106201	£14.95
Audio DDS Oscillator PCB	01110201	£5.95
Audio DDS Oscillator rotary encoder	01110201-ENC	6.95
Programming Adaptor Board for Audio DDS Oscillator	01110202	£5.95
High-power Ultrasonic Cleaner main PCB	04105201	£14.95
High-power Ultrasonic Cleaner front-panel PCB	04105202	
Night Keeper Lighthouse PCB	08110201	£5.95

PROJECT	CODE	PRICE
AUGUST 2021		
Ol' Timer PCB	19104201	£11.95
Ol' Timer 8x8 RGB LED module using WS2812B	19104201-88	£8.95
Ol' Timer set of acrylic case pieces and spacer	19104201-ACR	£8.75
Ol' Timer DS3231 RTC IC wide SOIC-16	19104201-RTC	£5.95
Wideband Digital RF Power Meter	04106201	£9.75
Switchmode 78xx regulators ***(PACK of 5!)***	18105201	£7.95
Cool Beans SMAD display	CB-AUG21	£11.95
JULY 2021		
ATtiny816 Breakout / Dev Board w/ Capacitive Touch	24110181	£9.75
IR Remote Control Assistant (Jaycar version)	15005201	£8.95
IR Remote Control Assistant (Altronics version)	15005202	£8.95
PIC18F Development Board	PNM-JUL21	£12.95
Microphone Preamplifier	AO-JUL21	£11.95
JUNE 2021		
Roadies' Test Signal Generator (surface-mount ver.)	01005201	£8.95
Roadies' Test Signal Generator (through-hole version)	01005202	£9.95
Touchscreen Wide-range RCL Box (Resistor module)	04104201	£18.95
Touchscreen Wide-range RCL Box (Ind/Cap module)	04104202	
KickStart Part 3 – Gyrator-based Audio Filter	KS3-2021	£7.95
MAY 2021		
7-Band Equaliser (Mono)	01104201	£8.95
7-Band Equaliser (Stereo)	01104202	£10.95
Car Altimeter	05105201	£7.95
APRIL 2021		
Reflow Oven – DSP Active Crossover (CPU)	01106193	£19.95
Reflow Oven – DSP Active Crossover (Front panel)	01106195	
Reflow Oven – DSP Active Crossover (LCD)	01106196	
Frequency Reference Signal Distributor	CSE200103	£8.95
MARCH 2021		
Nutube Guitar Effects Pedal	01102201	£12.95
Programmable Thermal Regulator (Peltier Interface)	21109181	£18.95
Programmable Thermal Regulator (Peltier Driver)	21109182	
Tunable HF Preamp	CSE190502	£8.95
FEBRUARY 2021		
4G Remote Monitoring	27111191	£9.95
JANUARY 2021		
Nutube Valve Preamplifier	01112191	£12.95
Arduino DCC Controller	09207181	£10.95
DECEMBER 2020		
Pseudo-Random Sequence Generator	16106191	£7.95
Clever Charger	14107191	£11.95
PE Theremin Amplifier	AO-1220-01	£8.95
NOVEMBER 2020		
LED Christmas Tree (1 off)	16107181-1	£6.95
LED Christmas Tree (4 off)	16107181-2	£14.95
LED Christmas Tree (12 off)	16107181-3	£24.95
LED Christmas Tree (20 off)	16107181-4	£34.95
USB/SPI Interface Board	16107182	£5.95
45V/8A Power Supply PCB *plus* acrylic spacer	18111181	£14.95
45V/8A Power Supply five-way display bezel	18111181-BZ	£3.95
Five-way LCD Panel Meter/Display	18111182	£7.95
OCTOBER 2020		
Digital Audio Millivoltmeter	04108191	£9.95
Precision Signal Amplifier	04107191	£7.95

Silicon Chip kits & parts for projects in this issue (posted from Australia):

Micromite Explore-40 kit	https://siliconchip.au/Shop/20/6991	AUD $42 (about £21) + P&P
Micromite Explore-40 audio breakout PCB with connectors	https://siliconchip.au/Shop/8/7326	AUD $3 (about £1.50) + P&P
FlexiDice kit	https://siliconchip.au/Shop/20/7361	AUD $36 (about £18) + P&P

20% VAT is included in the above prices (it will be removed for customers outside the UK). Postage & packing cost is typically around £12-15. Please allow 1-2 weeks for delivery (longer in some cases).

CLASSIFIED ADVERTISING

Practical Electronics

If you want your advertisements to be seen by the largest readership at the most economical price then our classified page offers excellent value.

The rate for semi-display space is £10 per centimetre high, with a minimum height of 2·5cm. All semi-display advertisements have a width of 5.5cm. The prepaid rate for classified adverts is 40p per word (minimum 12 words).

For rates, further information on display and classified advertising or to book an ad, please email us at **pe@pemag.au**

Practical Electronics reaches more UK readers than any other UK monthly hobby electronics magazine. Our sales figures prove it.

We have been the leading monthly magazine in this market for more than thirty years.

BOWOOD ELECTRONICS LTD
Suppliers of Electronic Components
www.bowood-electronics.co.uk
Unit 10, Boythorpe Business Park, Dock Walk, Chesterfield,
Derbyshire S40 2QR. Sales: 01246 200 222
Send large letter stamp for Catalogue

Electrical Industries Charity (EIC)
We help people working in the electrical, electronics and energy community as well as their family members and retirees.
We use workplace programmes that give the industry access to financial grants and a comprehensive range of free and confidential services.
www.electricalcharity.org
electrical industries charity

COAST ELECTRONICS
BREAKOUTS-COMPONENTS-
CONTRACT DESIGN-3D PRINTER PARTS-
MUSICAL-MICROCONTROLLERS
WWW.COASTELECTRONICS.CO.UK

MISCELLANEOUS

VALVES AND ALLIED COMPONENTS?
For free stock list and/or advice, please contact me: geoffdavies337@gmail.com Telephone: 01788 574774

PIC DEVELOPMENT KITS, DTMF kits and modules, CTCSS Encoder and Decoder/Display kits.
Visit www.cstech.co.uk

COMPONENTS AS JOB LOT
Large stock. Transistors, Diodes, Logic, Memory and Analogue parts. 1000s of passive items. Catalogued and sorted. 'New Old Stock' in racks.
Email: rojobarn@gmail.com

Andrew Kenny – Qualified Patent Agent
EPO UKIPO USPTO
Circuits Electric Machinery Mechatronics
Web: www.akennypatentm.com
Email: Enquiries@akennypatentm.com
Tel: 0789 606 9725

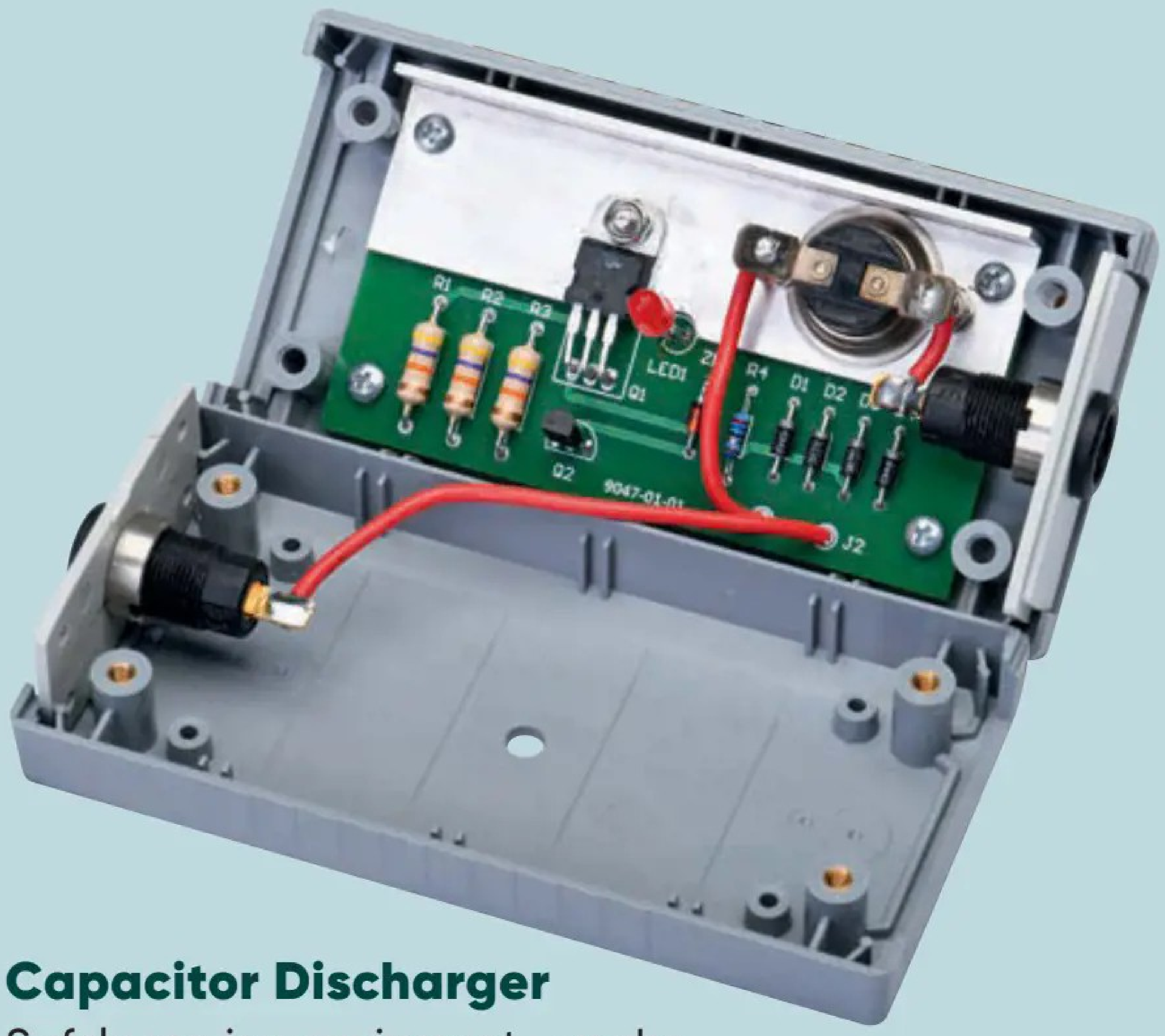
ADVERTISING INDEX

AOSHOP 71
BACK ISSUE PDFs. 21
BASIC MICRO 43
ELECTRONICS TEACH-IN BUNDLES 53
ENIGMA COMPONENTS SHOP 39
MICROCHIP TECHNOLOGY . . . . Outer back cover
PCBWAY. . . . . Inner front cover
SILICON CHIP MAGAZINE. 13
TAG-CONNECT 39

Advertising mailing address
Electron Publishing Australia
PO Box 194
Matraville NSW 2036
Australia
Email pe@pemag.au
Web www.electronpublishing.com

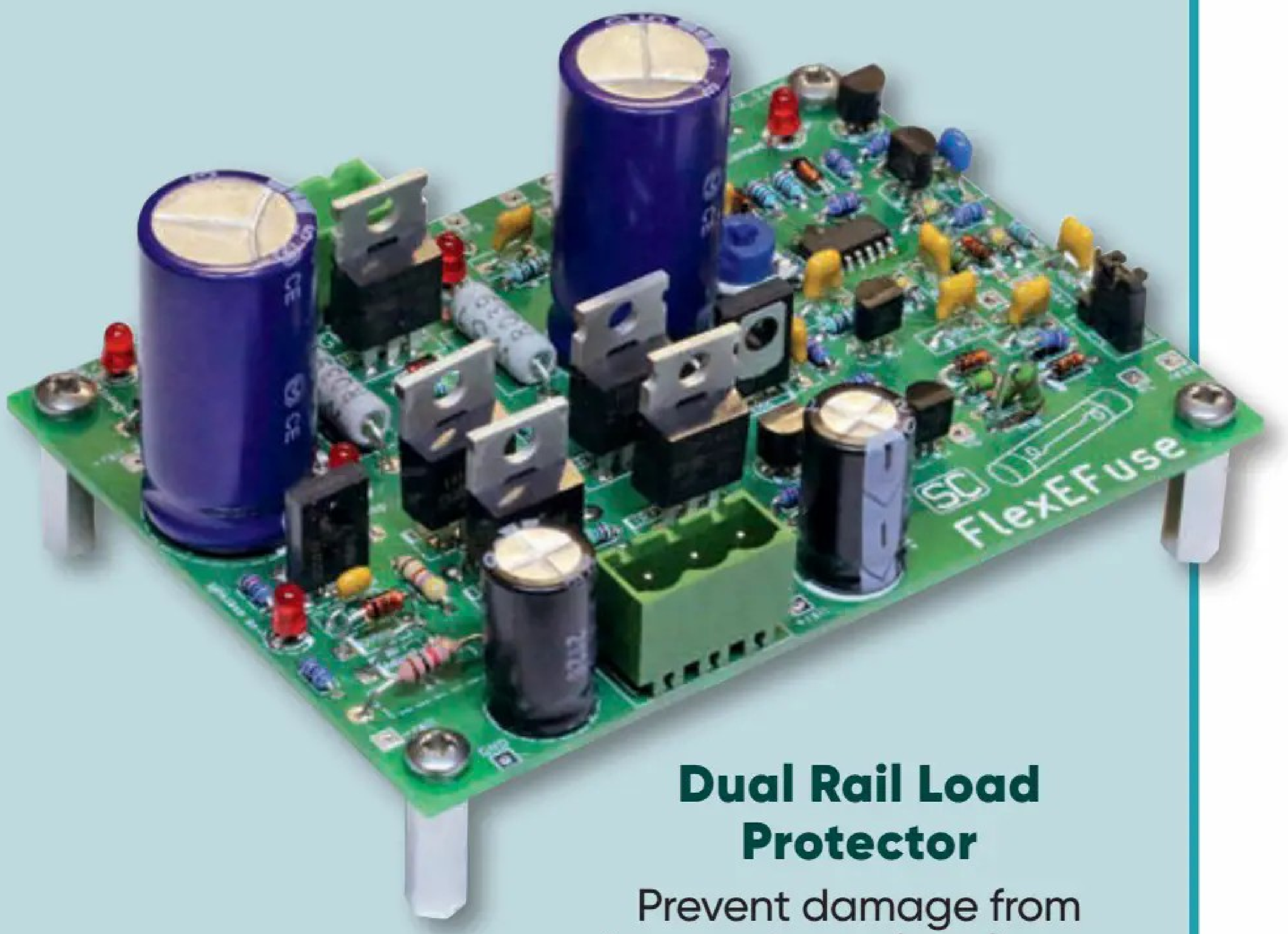
For editorial contact details, see page 2.

Next Month – in the September issue



Capacitor Discharger

Safely service equipment – make sure those capacitors are discharged first!



Dual Rail Load Protector

Prevent damage from incorrect supply voltages or excessive current

Compact Hi-Fi Headphone Amp

Easy to build, with great performance at a low cost.

On sale 5th of August, 2025

Content may be subject to change

Practical Electronics

Subscriptions hotline

To subscribe or renew online, paying by credit/debit card or PayPal, please visit our shop at: www.electronpublishing.com

For those wishing to pay by direct debit, cheque or over the telephone, please call 'Select Publisher Services' on 01202 087631, or send a cheque (payable to: '**Select Publisher Services Ltd**') with your details to:

Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH, United Kingdom

Remember, we print the date of the last issue of your current subscription in a box on the address sheet that comes with your copy.

Digital subscribers, please call 07918 614662 or visit: www.electronpublishing.com

Published on approximately the first Thursday of each month by Silicon Chip Publications, PO Box 194 Matraville NSW 2036 Australia. Printed in England by The Manson Group, Reynolds House, 8 Porters Wood, Valley Road Industrial Estate, St Albans AL3 6PZ. Distributed by Seymour, 86 Newman St, London W1T 3EX. Subscriptions UK: £33.99 (6mo); £59.99 (12mo.); £114.99 (2yrs). EUROPE: airmail service, £38.99 (6mo); £69.99 (12mo); £129.99 (2yrs). REST OF THE WORLD: airmail service, £46.99 (6mo); £84.99 (12mo); £164.99 (2yrs). Make payments to 'Practical Electronics', Practical Electronics Subscriptions, PO Box 6337, Bournemouth BH1 9EH United Kingdom. Email: pesubs@selectps.com. PRACTICAL ELECTRONICS is sold subject to the conditions that it shall not, without the written consent of the Publishers first having been given, be lent, resold, hired out or otherwise disposed of by way of Trade at more than the recommended selling price on the cover, and that it shall not be lent, resold, hired out or otherwise disposed of in a mutilated condition or in any unauthorised cover by way of Trade or affixed to or as part of any publication or advertising, literary or pictorial matter.

Did you know our online shop now sells the current issue of *PE* for £5.99 inc. p&p?

Practical Electronics
The UK's premier electronics, computing and maker magazine
Circuit Surgery
Getting our DSP board's ADC & DAC working together
Audio Out
Minimising distortion in balanced input transformers
Precision Electronics
Choosing & using voltage references
Explore-40 MMBasic in a Pico form factor
8-Channel IR Remote Relay
FlexiDice
PLUS!
Techno Talk – finding chip data & pseudo intelligence
Cool Beans – ultimate switch debouncing and RNG
MIPI's new I3C serial standard can replace both I²C & SPI
Aug 2025 £5.99
www.electronpublishing.com
@practicalelec
practicalelectronics
Practical Electronics
computing and maker magazine
Precision Electronics
Getting the most out of your ADC in precision applications
180-230V DC Motor Speed Controller with feedback
Jul 2025 £5.99
@practicalelec
practicalelectronics

You read that right! We now sell the current issue of your favourite electronics magazine for exactly the same price as in the High Street, but we deliver it straight to your door – and for UK addresses we pay the postage. No need to journey into town to queue outside the newsagent. Just go to our website, set up an account in 30 seconds, order your magazine and we'll do the rest.

www.electronpublishing.com

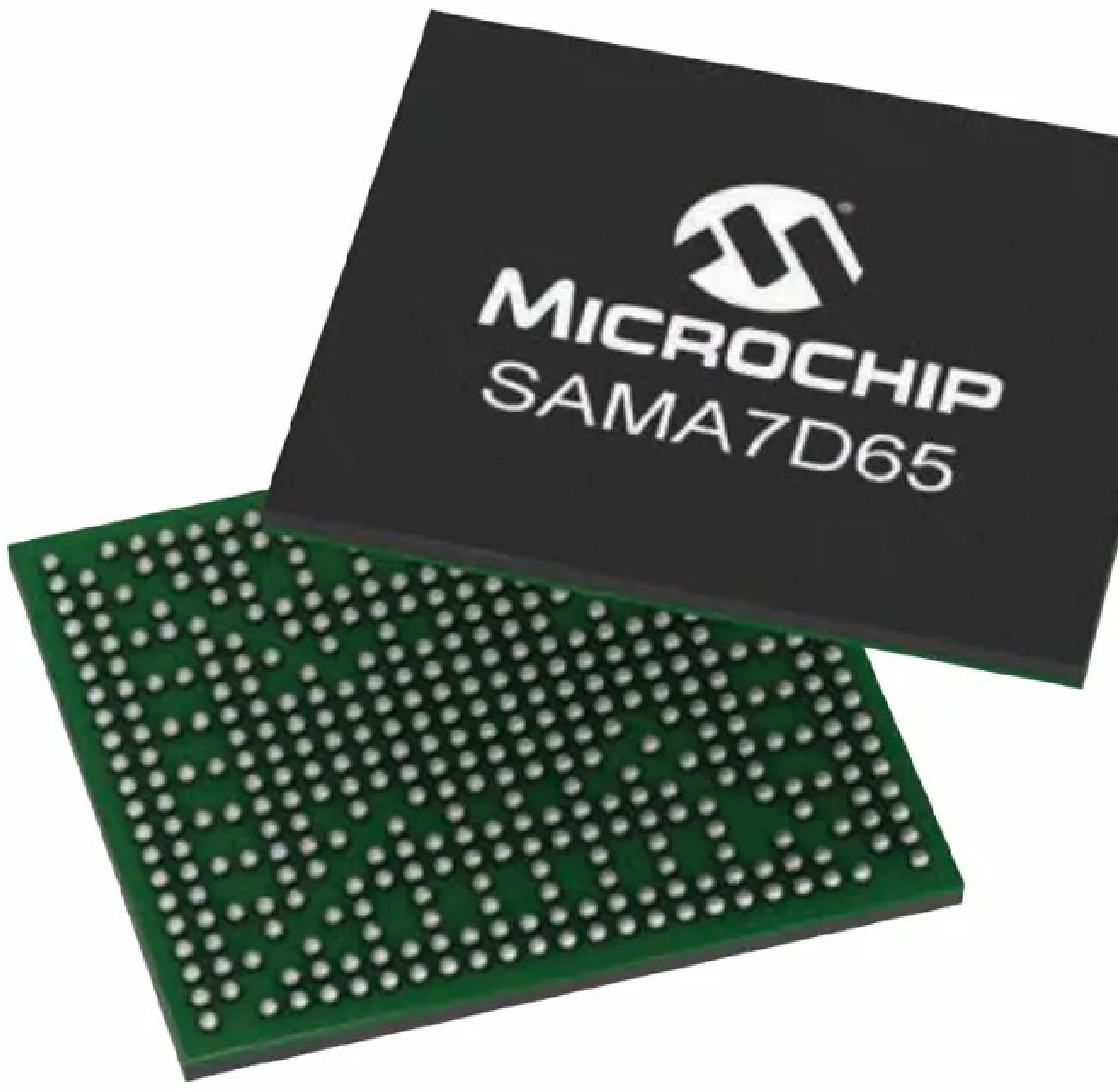
SAMA7D65 MPUs: Advanced Graphics, Connectivity

High-performance graphics and connectivity for HMI applications

SAMA7D65 MPUs are designed for advanced Human-Machine Interface (HMI) and connectivity applications, featuring a 1 GHz Arm Cortex-A7 core. Available in both System-in-Package (SiP) with 2 Gb DDR3L and System-on-Chip (SoC) configurations, these MPUs offer high-performance graphics and connectivity features. The series supports dual Gigabit Ethernet with Time Sensitive Networking (TSN) for precise synchronization and low-latency communication, making it ideal for industrial, medical and transportation markets. The SAMA7D65 MPUs are supported by Microchip Graphics Suite, MPLAB Harmony v3 and Linux, with a Curiosity Development Board available for evaluation.

Key Features

- 1 GHz Arm Cortex-A7 core with advanced graphics (LVDS, MIPI DSI, 2D GPU)
- Available in SiP with 2 Gb DDR3L and SoC configurations
- Early System on Module (SOM) samples available with 2Gb DDR3L 8Gb NAND
- Dual Gigabit Ethernet with TSN support for real-time systems
- Energy-efficient design with low-voltage DDR3L memory
- Supported by Microchip Graphics Suite, MPLAB Harmony v3 and Linux



Battery
100%

microchip.com/SAMAD765MPU

The Microchip name and logo and the Microchip logo are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries. All other trademarks are the property of their registered owners.
© 2025 Microchip Technology Inc. All rights reserved.